Euler's Polyhedron Formula

Lawrence C. Paulson

September 22, 2023

Abstract

Euler stated in 1752 that every convex polyhedron satisfied the formula $V - E + F = 2$ where V, E and F are the numbers of its vertices, edges, and faces. For three dimensions, the well-known proof involves removing one face and then flattening the remainder to form a planar graph, which then is iteratively transformed to leave a single triangle. The history of that proof is extensively discussed and elaborated by Imre Lakatos [1], leaving one finally wondering whether the theorem even holds. The formal proof provided here has been ported from HOL Light, where it is credited to Lawrence [2]. The proof generalises Euler's observation from solid polyhedra to convex polytopes of arbitrary dimension.

Contents

Acknowledgements The author was supported by the ERC Advanced Grant ALEXANDRIA (Project 742178) funded by the European Research Council.

1 Library Extras

For adding to the repository

theory *Library-Extras* **imports**
  HOL−Analysis.Polytope

begin

2 Preliminaries

lemma *Inter-over-Union*:
  $\bigcap \{\bigcup (\mathcal{F}\ x) \mid x.\ x \in S\} = \bigcup \{\bigcap (G\ `\ S) \mid G.\ \forall x{\in}S.\ G\ x \in \mathcal{F}\ x\}$
proof −
  have $\bigwedge x.\ \forall s{\in}S.\ \exists X \in \mathcal{F}\ s.\ x \in X \Longrightarrow \exists G.\ (\forall x{\in}S.\ G\ x \in \mathcal{F}\ x) \wedge (\forall s{\in}S.\ x \in G\ s)$
    by *metis*
  then show *?thesis*
    by (*auto simp flip*: *all-simps ex-simps*)
qed

lemmas *closure-Int-convex* = *convex-closure-inter-two*

lemmas *span-not-UNIV-orthogonal* = *span-not-univ-orthogonal*

lemma *convex-closure-rel-interior-Int*:
  assumes $\bigwedge S.\ S{\in}\mathcal{F} \Longrightarrow$ *convex* (S :: *'n::euclidean-space set*)
    and $\bigcap$(*rel-interior* ‘ $\mathcal{F}$) ≠ {}
  shows $\bigcap$(*closure* ‘ $\mathcal{F}$) ⊆ *closure* ($\bigcap$(*rel-interior* ‘ $\mathcal{F}$))
proof −
  obtain x **where** x: $\forall S{\in}\mathcal{F}.\ x \in$ *rel-interior* S
    using *assms* **by** *auto*
  show *?thesis*
  proof
    fix y
    assume y: $y \in \bigcap$ (*closure* ‘ $\mathcal{F}$)
    show $y \in$ *closure* ($\bigcap$(*rel-interior* ‘ $\mathcal{F}$))
    proof (*cases* $y{=}x$)
      case *True*
      with *closure-subset* x **show** *?thesis*
        by *fastforce*
    next
      case *False*
      { **fix** ε :: *real*
        assume e: $\varepsilon > 0$
        define $e1$ **where** $e1 = \min 1\ (\varepsilon/\mathit{norm}\ (y - x))$
        then have $e1$: $e1 > 0$ $e1 \le 1$ $e1 * \mathit{norm}\ (y - x) \le \varepsilon$
          using ‹$y \neq x$› ‹$\varepsilon > 0$› *le-divide-eq*[*of* $e1\ \varepsilon$ *norm* $(y - x)$]
          by *simp-all*

define z **where** $z = y - e1 *_R (y - x)$
{
fix S
assume $S \in \mathcal{F}$
then have $z \in$ *rel-interior* S
using *rel-interior-closure-convex-shrink*[*of* S x y *e1*] *assms* x y *e1* *z-def*
by *auto*
}
then have $*$: $z \in \bigcap($*rel-interior* ‘ $\mathcal{F})$
by *auto*
have $\exists x \in \bigcap$ (*rel-interior* ‘ $\mathcal{F}$). *dist* x $y \le \varepsilon$
using ‹$y \neq x$› *z-def* $*$ *e1* *e* *dist-norm*[*of* z y]
by *force*
} **then**
show *?thesis*
by (*auto simp*: *closure-approachable-le*)
qed
qed
qed

lemma *closure-Inter-convex*:
fixes $\mathcal{F}$ :: $'n$::*euclidean-space set set*
assumes $\bigwedge S.\ S \in \mathcal{F} \Longrightarrow$ *convex* S **and** $\bigcap($*rel-interior* ‘ $\mathcal{F}) \neq \{\}$
shows *closure*$(\bigcap \mathcal{F}) = \bigcap($*closure* ‘ $\mathcal{F})$
proof $-$
have $\bigcap($*closure* ‘ $\mathcal{F}) \le$ *closure* $(\bigcap($*rel-interior* ‘ $\mathcal{F}))$
by (*meson assms convex-closure-rel-interior-Int*)
moreover
have *closure* $(\bigcap($*rel-interior* ‘ $\mathcal{F})) \subseteq$ *closure* $(\bigcap \mathcal{F})$
using *rel-interior-inter-aux closure-mono*[*of* $\bigcap($*rel-interior* ‘ $\mathcal{F})$ $\bigcap \mathcal{F}$]
by *auto*
ultimately show *?thesis*
using *closure-Int*[*of* $\mathcal{F}$] **by** *blast*
qed

lemma *closure-Inter-convex-open*:
$(\bigwedge S$::$'n$::*euclidean-space set*. $S \in \mathcal{F} \Longrightarrow$ *convex* $S \wedge$ *open* $S)$
$\Longrightarrow$ *closure*$(\bigcap \mathcal{F}) = ($*if* $\bigcap \mathcal{F} = \{\}$ *then* $\{\}$ *else* $\bigcap($*closure* ‘ $\mathcal{F}))$
by (*simp add*: *closure-Inter-convex rel-interior-open*)

lemma *empty-interior-subset-hyperplane-aux*:
fixes S :: $'a$::*euclidean-space set*
assumes *convex* S $0 \in S$ **and** *empty-int*: *interior* $S = \{\}$
shows $\exists a\ b.\ a \neq 0 \wedge S \subseteq \{x.\ a \cdot x = b\}$
proof $-$
have *False* **if** $\bigwedge a.\ a = 0 \vee (\forall b.\ \exists T \in S.\ a \cdot T \neq b)$
proof $-$
have *rel-int*: *rel-interior* $S \neq \{\}$

using *assms rel-interior-eq-empty* **by** *auto*
moreover
have *dim S* $\neq$ *dim* (*UNIV*::*'a set*)
by (*metis aff-dim-zero affine-hull-UNIV* ‹*0* $\in$ *S*› *dim-UNIV empty-int hull-inc rel-int rel-interior-interior*)
then obtain *a* **where** *a* $\neq$ *0* **and** *a*: *span S* $\subseteq$ {*x. a* $\cdot$ *x* = *0*}
using *lowdim-subset-hyperplane*
by (*metis dim-UNIV dim-subset-UNIV order-less-le*)
have *span UNIV* = *span S*
by (*metis span-base span-not-UNIV-orthogonal that*)
then have *UNIV* $\subseteq$ *affine hull S*
by (*simp add*: ‹*0* $\in$ *S*› *hull-inc affine-hull-span-0*)
ultimately show *False*
using ‹*rel-interior S* $\neq$ {}› *empty-int rel-interior-interior* **by** *blast*
qed
then show *?thesis*
by *blast*
qed

lemma *empty-interior-subset-hyperplane*:
fixes *S* :: *'a*::*euclidean-space set*
assumes *convex S* **and** *int*: *interior S* = {}
obtains *a b* **where** *a* $\neq$ *0 S* $\subseteq$ {*x. a* $\cdot$ *x* = *b*}
proof (*cases S* = {})
case *True*
then show *?thesis*
using *that* **by** *blast*
next
case *False*
then obtain *u* **where** *u* $\in$ *S*
by *blast*
have $\exists$ *a b. a* $\neq$ *0* $\wedge$ (λ*x. x* − *u*) ‘ *S* $\subseteq$ {*x. a* $\cdot$ *x* = *b*}
proof (*rule empty-interior-subset-hyperplane-aux*)
show *convex* ((λ*x. x* − *u*) ‘ *S*)
using ‹*convex S*› **by** *force*
show *0* $\in$ (λ*x. x* − *u*) ‘ *S*
by (*simp add*: ‹*u* $\in$ *S*›)
show *interior* ((λ*x. x* − *u*) ‘ *S*) = {}
by (*simp add*: *int interior-translation-subtract*)
qed
then obtain *a b* **where** *a* $\neq$ *0* **and** *ab*: (λ*x. x* − *u*) ‘ *S* $\subseteq$ {*x. a* $\cdot$ *x* = *b*}
by *metis*
then have *S* $\subseteq$ {*x. a* $\cdot$ *x* = *b* + (*a* $\cdot$ *u*)}
using *ab* **by** (*auto simp*: *algebra-simps*)
then show *?thesis*
using ‹*a* $\neq$ *0*› *that* **by** *auto*
qed

lemma *aff-dim-psubset*:

$(\textit{affine hull } S) \subset (\textit{affine hull } T) \Longrightarrow \textit{aff-dim } S < \textit{aff-dim } T$
by (*metis aff-dim-affine-hull aff-dim-empty aff-dim-subset affine-affine-hull affine-dim-equal order-less-le*)

lemma *aff-dim-eq-full-gen*:
$S \subseteq T \Longrightarrow (\textit{aff-dim } S = \textit{aff-dim } T \longleftrightarrow \textit{affine hull } S = \textit{affine hull } T)$
by (*smt* (*verit*, *del-insts*) *aff-dim-affine-hull2 aff-dim-psubset hull-mono psubsetI*)

lemma *aff-dim-eq-full*:
fixes $S :: 'n::\textit{euclidean-space set}$
shows $\textit{aff-dim } S = (\textit{DIM}('n)) \longleftrightarrow \textit{affine hull } S = \textit{UNIV}$
by (*metis aff-dim-UNIV aff-dim-affine-hull affine-hull-UNIV*)

3 Conic sets and conic hull

definition $\textit{conic} :: 'a::\textit{real-vector set} \Rightarrow \textit{bool}$
where $\textit{conic } S \equiv \forall x\ c.\ x \in S \longrightarrow 0 \leq c \longrightarrow (c *_R x) \in S$

lemma *conicD*: $[\![\textit{conic } S;\ x \in S;\ 0 \leq c]\!] \Longrightarrow (c *_R x) \in S$
by (*meson conic-def*)

lemma *subspace-imp-conic*: $\textit{subspace } S \Longrightarrow \textit{conic } S$
by (*simp add*: *conic-def subspace-def*)

lemma *conic-empty* [*simp*]: *conic* {}
using *conic-def* **by** *blast*

lemma *conic-UNIV*: *conic UNIV*
by (*simp add*: *conic-def*)

lemma *conic-Inter*: $(\bigwedge S.\ S \in \mathcal{F} \Longrightarrow \textit{conic } S) \Longrightarrow \textit{conic}(\bigcap \mathcal{F})$
by (*simp add*: *conic-def*)

lemma *conic-linear-image*:
$[\![\textit{conic } S;\ \textit{linear } f]\!] \Longrightarrow \textit{conic}(f \text{ ` } S)$
by (*smt* (*verit*) *conic-def image-iff linear.scaleR*)

lemma *conic-linear-image-eq*:
$[\![\textit{linear } f;\ \textit{inj } f]\!] \Longrightarrow \textit{conic } (f \text{ ` } S) \longleftrightarrow \textit{conic } S$
by (*smt* (*verit*) *conic-def conic-linear-image inj-image-mem-iff linear-cmul*)

lemma *conic-mul*: $[\![\textit{conic } S;\ x \in S;\ 0 \leq c]\!] \Longrightarrow (c *_R x) \in S$
using *conic-def* **by** *blast*

lemma *conic-conic-hull*: *conic*(*conic hull S*)
by (*metis* (*no-types*, *lifting*) *conic-Inter hull-def mem-Collect-eq*)

lemma *conic-hull-eq*: $(\textit{conic hull } S = S) \longleftrightarrow \textit{conic } S$
by (*metis conic-conic-hull hull-same*)

lemma *conic-hull-UNIV* [*simp*]: *conic hull UNIV* = *UNIV*
  by *simp*

lemma *conic-negations*: *conic S* $\Longrightarrow$ *conic* (*image uminus S*)
  by (*auto simp*: *conic-def image-iff*)

lemma *conic-span* [*iff*]: *conic*(*span S*)
  by (*simp add*: *subspace-imp-conic*)

lemma *conic-hull-explicit*:
  conic hull S = $\{c *_R x \mid c\ x.\ 0 \le c \land x \in S\}$
proof (*rule hull-unique*)
  show $S \subseteq \{c *_R x \mid c\ x.\ 0 \le c \land x \in S\}$
    by (*metis* (*no-types*) *cone-hull-expl hull-subset*)
  show *conic* $\{c *_R x \mid c\ x.\ 0 \le c \land x \in S\}$
    using *mult-nonneg-nonneg* **by** (*force simp*: *conic-def*)
qed (*auto simp*: *conic-def*)

lemma *conic-hull-as-image*:
  conic hull S = ($\lambda z.$ *fst z* $*_R$ *snd z*) ‘ ($\{0..\} \times S$)
  by (*force simp*: *conic-hull-explicit*)

lemma *conic-hull-linear-image*:
  linear f $\Longrightarrow$ *conic hull f* ‘ *S* = *f* ‘ (*conic hull S*)
  by (*force simp*: *conic-hull-explicit image-iff set-eq-iff linear-scale*)

lemma *conic-hull-image-scale*:
  assumes $\bigwedge x.\ x \in S \Longrightarrow 0 < c\ x$
  shows *conic hull* ($\lambda x.\ c\ x *_R x$) ‘ *S* = *conic hull S*
proof
  show *conic hull* ($\lambda x.\ c\ x *_R x$) ‘ $S \subseteq$ *conic hull S*
  proof (*rule hull-minimal*)
    show ($\lambda x.\ c\ x *_R x$) ‘ $S \subseteq$ *conic hull S*
      using *assms conic-hull-explicit* **by** *fastforce*
  qed (*simp add*: *conic-conic-hull*)
  show *conic hull* $S \subseteq$ *conic hull* ($\lambda x.\ c\ x *_R x$) ‘ *S*
  proof (*rule hull-minimal*)
    { fix *x*
      assume $x \in S$
      then have x = *inverse*($c\ x$) $*_R c\ x *_R x$
        using *assms* **by** *fastforce*
      then have $x \in$ *conic hull* ($\lambda x.\ c\ x *_R x$) ‘ *S*
          by (*smt* (*verit*, *best*) ‹$x \in S$› *assms conic-conic-hull conic-mul hull-inc image-eqI inverse-nonpositive-iff-nonpositive*)
    }
    then show $S \subseteq$ *conic hull* ($\lambda x.\ c\ x *_R x$) ‘ *S* **by** *auto*
  qed (*simp add*: *conic-conic-hull*)
qed

```
lemma convex-conic-hull:
  assumes convex S
  shows convex (conic hull S)
proof −
  { fix c x d y and u :: real
    assume §: (0::real) ≤ c x ∈ S (0::real) ≤ d y ∈ S 0 ≤ u u ≤ 1
    have ∃ c'' x''. ((1 − u) ∗ c) ∗R x + (u ∗ d) ∗R y = c'' ∗R x'' ∧ 0 ≤ c'' ∧ x'' ∈ S
    proof (cases (1 − u) ∗ c = 0)
      case True
      with ‹0 ≤ d› ‹y ∈ S›‹0 ≤ u›
      show ?thesis by force
    next
      case False
      define ξ where ξ ≡ (1 − u) ∗ c + u ∗ d
      have ∗: c ∗ u ≤ c
        by (simp add: § mult-left-le)
      have ξ > 0
        using False § by (smt (verit, best) ξ-def split-mult-pos-le)
      then have ∗∗: c + d ∗ u = ξ + c ∗ u
        by (simp add: ξ-def mult.commute right-diff-distrib')
      show ?thesis
      proof (intro exI conjI)
        show 0 ≤ ξ
          using ‹0 < ξ› by auto
        show ((1 − u) ∗ c) ∗R x + (u ∗ d) ∗R y = ξ ∗R (((1 − u) ∗ c / ξ) ∗R x + (u ∗ d / ξ) ∗R y)
          using ‹ξ > 0› by (simp add: algebra-simps diff-divide-distrib)
        show ((1 − u) ∗ c / ξ) ∗R x + (u ∗ d / ξ) ∗R y ∈ S
          using ‹0 < ξ›
          by (intro convexD [OF assms]) (auto simp: § field-split-simps ∗ ∗∗)
      qed
    qed
  }
  then show ?thesis
    by (auto simp add: conic-hull-explicit convex-alt)
qed

lemma conic-halfspace-le: conic {x. a · x ≤ 0}
  by (auto simp: conic-def mult-le-0-iff)

lemma conic-halfspace-ge: conic {x. a · x ≥ 0}
  by (auto simp: conic-def mult-le-0-iff)

lemma conic-hull-empty [simp]: conic hull {} = {}
  by (simp add: conic-hull-eq)

lemma conic-contains-0: conic S ⟹ (0 ∈ S ⟷ S ≠ {})
```

by (*simp add*: *Convex.cone-def cone-contains-0 conic-def*)

lemma *conic-hull-eq-empty*: *conic hull* $S = \{\} \longleftrightarrow (S = \{\})$
using *conic-hull-explicit* **by** *fastforce*

lemma *conic-sums*: $[\![$*conic* S; *conic* $T]\!] \Longrightarrow$ *conic* $(\bigcup x \in S. \bigcup y \in T. \{x + y\})$
by (*simp add*: *conic-def*) (*metis scaleR-right-distrib*)

lemma *conic-Times*: $[\![$*conic* S; *conic* $T]\!] \Longrightarrow$ *conic*$(S \times T)$
by (*auto simp*: *conic-def*)

lemma *conic-Times-eq*:
conic$(S \times T) \longleftrightarrow S = \{\} \vee T = \{\} \vee$ *conic* $S \wedge$ *conic* T (**is** *?lhs* = *?rhs*)
proof
show *?lhs* $\Longrightarrow$ *?rhs*
by (*force simp*: *conic-def*)
show *?rhs* $\Longrightarrow$ *?lhs*
by (*force simp*: *conic-Times*)
qed

lemma *conic-hull-0* [*simp*]: *conic hull* $\{0\} = \{0\}$
by (*simp add*: *conic-hull-eq subspace-imp-conic*)

lemma *conic-hull-contains-0* [*simp*]: $0 \in$ *conic hull* $S \longleftrightarrow (S \neq \{\})$
by (*simp add*: *conic-conic-hull conic-contains-0 conic-hull-eq-empty*)

lemma *conic-hull-eq-sing*:
conic hull $S = \{x\} \longleftrightarrow S = \{0\} \wedge x = 0$
proof
show *conic hull* $S = \{x\} \Longrightarrow S = \{0\} \wedge x = 0$
by (*metis conic-conic-hull conic-contains-0 conic-def conic-hull-eq hull-inc insert-not-empty singleton-iff*)
qed *simp*

lemma *conic-hull-Int-affine-hull*:
assumes $T \subseteq S$ $0 \notin$ *affine hull* S
shows (*conic hull* T) $\cap$ (*affine hull* S) = T
proof –
have *TaffS*: $T \subseteq$ *affine hull* S
using ‹$T \subseteq S$› *hull-subset* **by** *fastforce*
moreover
{ **fix** c x
assume $c *_R x \in$ *affine hull* S
and $0 \leq c$
and $x \in T$
have $c *_R x \in T$
proof (*cases* c=*1*)
case *True*
then show *?thesis*

```
      by (simp add: ‹x ∈ T›)
    next
      case False
      then have x /R (1 − c) = x + (c ∗ inverse (1 − c)) ∗R x
       by (smt (verit, ccfv-SIG) diff-add-cancel mult.commute real-vector-affinity-eq
scaleR-collapse scaleR-scaleR)
      then have 0 = inverse(1 − c) ∗R c ∗R x + (1 − inverse(1 − c)) ∗R x
        by (simp add: algebra-simps)
      then have 0 ∈ affine hull S
         by (smt (verit) ‹c ∗R x ∈ affine hull S› ‹x ∈ T› affine-affine-hull TaffS
in-mono mem-affine)
      then show ?thesis
        using assms by auto
    qed }
  then have conic hull T ∩ affine hull S ⊆ T
    by (auto simp: conic-hull-explicit)
  ultimately show ?thesis
    by (auto simp: hull-inc)
qed


lemma open-in-subset-relative-interior:
  fixes S :: 'a::euclidean-space set
  shows openin (top-of-set (affine hull T)) S ⟹ (S ⊆ rel-interior T) = (S ⊆ T)
  by (meson order.trans rel-interior-maximal rel-interior-subset)


lemma conic-hull-eq-span-affine-hull:
  fixes S :: 'a::euclidean-space set
  assumes 0 ∈ rel-interior S
  shows conic hull S = span S ∧ conic hull S = affine hull S
proof −
  obtain ε where ε>0 and ε: cball 0 ε ∩ affine hull S ⊆ S
    using assms mem-rel-interior-cball by blast
  have ∗: affine hull S = span S
    by (meson affine-hull-span-0 assms hull-inc mem-rel-interior-cball)
  moreover
  have conic hull S ⊆ span S
    by (simp add: hull-minimal span-superset)
  moreover
    { fix x
    assume x ∈ affine hull S
    have x ∈ conic hull S
    proof (cases x=0)
      case True
      then show ?thesis
        using ‹x ∈ affine hull S› by auto
    next
      case False
```

then have $(\varepsilon\ /\ norm\ x) *_R x \in cball\ 0\ \varepsilon \cap affine\ hull\ S$
  using ‹$0 < \varepsilon$› ‹$x \in affine\ hull\ S$› $*$ *span-mul* **by** *fastforce*
then have $(\varepsilon\ /\ norm\ x) *_R x \in S$
  by (*meson* ε *subsetD*)
then have $\exists c\ xa.\ x = c *_R xa \wedge 0 \leq c \wedge xa \in S$
  by (*smt* (*verit*, *del-insts*) ‹$0 < \varepsilon$› *divide-nonneg-nonneg eq-vector-fraction-iff norm-eq-zero norm-ge-zero*)
then show *?thesis*
  by (*simp add*: *conic-hull-explicit*)
qed
}
then have $affine\ hull\ S \subseteq conic\ hull\ S$
  by *auto*
ultimately show *?thesis*
  by *blast*
qed

lemma *conic-hull-eq-span*:
fixes S :: *'a::euclidean-space set*
assumes $0 \in rel\text{-}interior\ S$
shows $conic\ hull\ S = span\ S$
by (*simp add*: *assms conic-hull-eq-span-affine-hull*)

lemma *conic-hull-eq-affine-hull*:
fixes S :: *'a::euclidean-space set*
assumes $0 \in rel\text{-}interior\ S$
shows $conic\ hull\ S = affine\ hull\ S$
using *assms conic-hull-eq-span-affine-hull* **by** *blast*

lemma *conic-hull-eq-span-eq*:
fixes S :: *'a::euclidean-space set*
shows $0 \in rel\text{-}interior(conic\ hull\ S) \longleftrightarrow conic\ hull\ S = span\ S$ (**is** $?lhs = ?rhs$)
proof
show $?lhs \Longrightarrow ?rhs$
  by (*metis conic-hull-eq-span conic-span hull-hull hull-minimal hull-subset span-eq*)
show $?rhs \Longrightarrow ?lhs$
  by (*metis rel-interior-affine subspace-affine subspace-span*)
qed

4 Closure of conic hulls

proposition *closedin-conic-hull*:
fixes S :: *'a::euclidean-space set*
assumes $compact\ T\ 0 \notin T\ T \subseteq S$
shows *closedin* (*top-of-set* (*conic hull S*)) (*conic hull T*)
proof −
have $**$: $compact\ (\{0..\} \times T \cap (\lambda z.\ fst\ z *_R snd\ z) -` K)$ (**is** *compact ?L*)
  if $K \subseteq (\lambda z.\ (fst\ z) *_R snd\ z)\ `\ (\{0..\} \times S)$ *compact K* **for** K
proof −

```
    obtain r where r > 0 and r: ⋀x. x ∈ K ⟹ norm x ≤ r
      by (metis ‹compact K› bounded-normE compact-imp-bounded)
    show ?thesis
      unfolding compact-eq-bounded-closed
    proof
      have bounded ({0..r / setdist{0}T} × T)
        by (simp add: assms(1) bounded-Times compact-imp-bounded)
      moreover
      { fix a b
        assume a *R b ∈ K and b ∈ T and 0 ≤ a
        have setdist {0} T ≠ 0
          using ‹b ∈ T› assms compact-imp-closed setdist-eq-0-closed by auto
        then have T0: setdist {0} T > 0
          using less-eq-real-def by fastforce
        then have a * setdist {0} T ≤ r
          by (smt (verit, ccfv-SIG) ‹0 ≤ a› ‹a *R b ∈ K› ‹b ∈ T› dist-0-norm
mult-mono′ norm-scaleR r setdist-le-dist singletonI)
        with T0 ‹r>0› have a ≤ r / setdist {0} T
          by (simp add: divide-simps)
      }
      then have ?L ⊆ ({0..r / setdist{0}T} × T) by auto
      ultimately show bounded ?L
        by (meson bounded-subset)
      show closed ?L
      proof (rule continuous-closed-preimage)
        show continuous-on ({0..} × T) (λz. fst z *R snd z)
          by (intro continuous-intros)
        show closed ({0::real..} × T)
          by (simp add: assms(1) closed-Times compact-imp-closed)
        show closed K
          by (simp add: compact-imp-closed that(2))
      qed
    qed
  qed
  show ?thesis
    unfolding conic-hull-as-image
  proof (rule proper-map)
    show compact ({0..} × T ∩ (λz. fst z *R snd z) −‘ K) (is compact ?L)
      if K ⊆ (λz. (fst z) *R snd z) ‘ ({0..} × S) compact K for K
    proof −
      obtain r where r > 0 and r: ⋀x. x ∈ K ⟹ norm x ≤ r
        by (metis ‹compact K› bounded-normE compact-imp-bounded)
      show ?thesis
        unfolding compact-eq-bounded-closed
      proof
        have bounded ({0..r / setdist{0}T} × T)
          by (simp add: assms(1) bounded-Times compact-imp-bounded)
        moreover
        { fix a b
```

```
          assume a *_R b ∈ K and b ∈ T and 0 ≤ a
          have setdist {0} T ≠ 0
            using ‹b ∈ T› assms compact-imp-closed setdist-eq-0-closed by auto
          then have T0: setdist {0} T > 0
            using less-eq-real-def by fastforce
          then have a * setdist {0} T ≤ r
             by (smt (verit, ccfv-SIG) ‹0 ≤ a› ‹a *_R b ∈ K› ‹b ∈ T› dist-0-norm
mult-mono' norm-scaleR r setdist-le-dist singletonI)
          with T0 ‹r>0› have a ≤ r / setdist {0} T
            by (simp add: divide-simps)
        }
        then have ?L ⊆ ({0..r / setdist{0}T} × T) by auto
        ultimately show bounded ?L
          by (meson bounded-subset)
        show closed ?L
        proof (rule continuous-closed-preimage)
          show continuous-on ({0..} × T) (λz. fst z *_R snd z)
            by (intro continuous-intros)
          show closed ({0::real..} × T)
            by (simp add: assms(1) closed-Times compact-imp-closed)
          show closed K
            by (simp add: compact-imp-closed that(2))
        qed
      qed
    qed
    show (λz. fst z *_R snd z) ` ({0::real..} × T) ⊆ (λz. fst z *_R snd z) ` ({0..} ×
S)
      using ‹T ⊆ S› by force
  qed auto
qed

lemma closed-conic-hull:
  fixes S :: 'a::euclidean-space set
  assumes 0 ∈ rel-interior S ∨ compact S ∧ 0 ∉ S
  shows   closed(conic hull S)
  using assms
proof
  assume 0 ∈ rel-interior S
  then show closed (conic hull S)
    by (simp add: conic-hull-eq-span)
next
  assume compact S ∧ 0 ∉ S
  then have closedin (top-of-set UNIV) (conic hull S)
    using closedin-conic-hull by force
  then show closed (conic hull S)
    by simp
qed

lemma conic-closure:
```

fixes *S* :: *'a::euclidean-space set*
shows *conic S* $\Longrightarrow$ *conic*(*closure S*)
by (*meson Convex.cone-def cone-closure conic-def*)

lemma *closure-conic-hull*:
fixes *S* :: *'a::euclidean-space set*
assumes *0* $\in$ *rel-interior S* $\vee$ *bounded S* $\wedge$ ~(*0* $\in$ *closure S*)
shows *closure*(*conic hull S*) = *conic hull* (*closure S*)
using *assms*
proof
assume *0* $\in$ *rel-interior S*
then show *closure* (*conic hull S*) = *conic hull closure S*
by (*metis closed-affine-hull closure-closed closure-same-affine-hull closure-subset conic-hull-eq-affine-hull subsetD subset-rel-interior*)
next
have $\bigwedge$*x. x* $\in$ *conic hull closure S* $\Longrightarrow$ *x* $\in$ *closure* (*conic hull S*)
by (*metis* (*no-types, opaque-lifting*) *closure-mono conic-closure conic-conic-hull subset-eq subset-hull*)
moreover
assume *bounded S* $\wedge$ *0* $\notin$ *closure S*
then have $\bigwedge$*x. x* $\in$ *closure* (*conic hull S*) $\Longrightarrow$ *x* $\in$ *conic hull closure S*
by (*metis closed-conic-hull closure-Un-frontier closure-closed closure-mono compact-closure hull-Un-subset le-sup-iff subsetD*)
ultimately show *closure* (*conic hull S*) = *conic hull closure S*
by *blast*
qed

lemma *faces-of-linear-image*:
$[\![$*linear f*; *inj f*$]\!]$ $\Longrightarrow$ {*T. T face-of* (*f* ` *S*)} = (*image f*) ` {*T. T face-of S*}
by (*smt* (*verit*) *Collect-cong face-of-def face-of-linear-image setcompr-eq-image subset-imageE*)

lemma *face-of-conic*:
assumes *conic S f face-of S*
shows *conic f*
unfolding *conic-def*
proof (*intro strip*)
fix *x* **and** *c::real*
assume *x* $\in$ *f* **and** *0* $\leq$ *c*
have *f*: $\bigwedge$*a b x.* $[\![$*a* $\in$ *S*; *b* $\in$ *S*; *x* $\in$ *f*; *x* $\in$ *open-segment a b*$]\!]$ $\Longrightarrow$ *a* $\in$ *f* $\wedge$ *b* $\in$ *f*
using ‹*f face-of S*› *face-ofD* **by** *blast*
show $c *_R x \in f$
proof (*cases x=0* $\vee$ *c=1*)
case *True*
then show *?thesis*
using ‹*x* $\in$ *f*› **by** *auto*
next
case *False*

with ‹$0 \leq c$› **obtain** d e **where** *de*: $0 \leq d$ $0 \leq e$ $d < 1$ $1 < e$ $d < e$ $(d = c \vee e = c)$
apply (*simp add*: *neq-iff*)
by (*metis gt-ex less-eq-real-def order-less-le-trans zero-less-one*)
then obtain [*simp*]: $c *_R x \in S$ $e *_R x \in S$ ‹$x \in S$›
using ‹$x \in f$› *assms conic-mul face-of-imp-subset* **by** *blast*
have $x \in$ *open-segment* $(d *_R x)$ $(e *_R x)$ **if** $c *_R x \notin f$
using *de False that*
apply (*simp add*: *in-segment*)
apply (*rule exI* [**where** $x=(1 - d) / (e - d)$])
apply (*simp add*: *field-simps*)
by (*smt* (*verit*, *del-insts*) *add-divide-distrib divide-self scaleR-collapse*)
then show *?thesis*
using ‹*conic S*› *f* [*of* $d *_R x$ $e *_R x$ x] *de* ‹$x \in f$›
by (*force simp*: *conic-def in-segment*)
qed
qed

lemma *extreme-point-of-conic*:
assumes *conic S* **and** *x*: *x extreme-point-of S*
shows $x = 0$
proof −
have $\{x\}$ *face-of S*
by (*simp add*: *face-of-singleton x*)
then have *conic*$\{x\}$
using *assms*(*1*) *face-of-conic* **by** *blast*
then show *?thesis*
by (*force simp*: *conic-def*)
qed

5 Convex cones and corresponding hulls

definition *convex-cone* :: $'a$::*real-vector set* $\Rightarrow$ *bool*
where *convex-cone* $\equiv \lambda S.\ S \neq \{\} \wedge$ *convex S* $\wedge$ *conic S*

lemma *convex-cone-iff*:
convex-cone S $\longleftrightarrow$
$0 \in S \wedge (\forall x \in S.\ \forall y \in S.\ x + y \in S) \wedge (\forall x \in S.\ \forall c{\geq}0.\ c *_R x \in S)$
by (*metis cone-def conic-contains-0 conic-def convex-cone convex-cone-def*)

lemma *convex-cone-add*: $[\![$*convex-cone S*; $x \in S$; $y \in S]\!] \Longrightarrow x{+}y \in S$
by (*simp add*: *convex-cone-iff*)

lemma *convex-cone-scaleR*: $[\![$*convex-cone S*; $0 \leq c$; $x \in S]\!] \Longrightarrow c *_R x \in S$
by (*simp add*: *convex-cone-iff*)

lemma *convex-cone-nonempty*: *convex-cone S* $\Longrightarrow S \neq \{\}$
by (*simp add*: *convex-cone-def*)

lemma *convex-cone-linear-image*:
convex-cone S $\wedge$ *linear f* $\Longrightarrow$ *convex-cone*(*f* ' *S*)
by (*simp add*: *conic-linear-image convex-cone-def convex-linear-image*)

lemma *convex-cone-linear-image-eq*:
$[\![$*linear f*; *inj f*$]\!]$ $\Longrightarrow$ (*convex-cone*(*f* ' *S*) $\longleftrightarrow$ *convex-cone S*)
by (*simp add*: *conic-linear-image-eq convex-cone-def*)

lemma *convex-cone-halfspace-ge*: *convex-cone* $\{x.\ a \cdot x \geq 0\}$
by (*simp add*: *convex-cone-iff inner-simps*(2))

lemma *convex-cone-halfspace-le*: *convex-cone* $\{x.\ a \cdot x \leq 0\}$
by (*simp add*: *convex-cone-iff inner-right-distrib mult-nonneg-nonpos*)

lemma *convex-cone-contains-0*: *convex-cone S* $\Longrightarrow$ $0 \in S$
using *convex-cone-iff* **by** *blast*

lemma *convex-cone-Inter*:
($\bigwedge S.\ S \in f \Longrightarrow$ *convex-cone S*) $\Longrightarrow$ *convex-cone*($\bigcap$ *f*)
by (*simp add*: *convex-cone-iff*)

lemma *convex-cone-convex-cone-hull*: *convex-cone*(*convex-cone hull S*)
by (*metis* (*no-types*, *lifting*) *convex-cone-Inter hull-def mem-Collect-eq*)

lemma *convex-convex-cone-hull*: *convex*(*convex-cone hull S*)
by (*meson convex-cone-convex-cone-hull convex-cone-def*)

lemma *conic-convex-cone-hull*: *conic*(*convex-cone hull S*)
by (*metis convex-cone-convex-cone-hull convex-cone-def*)

lemma *convex-cone-hull-nonempty*: *convex-cone hull S* $\neq \{\}$
by (*simp add*: *convex-cone-convex-cone-hull convex-cone-nonempty*)

lemma *convex-cone-hull-contains-0*: $0 \in$ *convex-cone hull S*
by (*simp add*: *convex-cone-contains-0 convex-cone-convex-cone-hull*)

lemma *convex-cone-hull-add*:
$[\![x \in$ *convex-cone hull S*; $y \in$ *convex-cone hull S*$]\!]$ $\Longrightarrow$ $x + y \in$ *convex-cone hull*
S
by (*simp add*: *convex-cone-add convex-cone-convex-cone-hull*)

lemma *convex-cone-hull-mul*:
$[\![x \in$ *convex-cone hull S*; $0 \leq c]\!]$ $\Longrightarrow$ $(c *_R x) \in$ *convex-cone hull S*
by (*simp add*: *conic-convex-cone-hull conic-mul*)

lemma *convex-cone-sums*:
$[\![$*convex-cone S*; *convex-cone T*$]\!]$ $\Longrightarrow$ *convex-cone* ($\bigcup x \in S.\ \bigcup y \in T.\ \{x + y\}$)
by (*simp add*: *convex-cone-def conic-sums convex-sums*)

lemma *convex-cone-Times*:
$[\![$*convex-cone S*; *convex-cone T*$]\!] \Longrightarrow$ *convex-cone*($S \times T$)
by (*simp add*: *conic-Times convex-Times convex-cone-def*)

lemma *convex-cone-Times-D1*: *convex-cone* ($S \times T$) $\Longrightarrow$ *convex-cone S*
by (*metis Times-empty conic-Times-eq convex-cone-def convex-convex-hull convex-hull-Times hull-same times-eq-iff*)

lemma *convex-cone-Times-eq*:
convex-cone($S \times T$) $\longleftrightarrow$ *convex-cone S* $\wedge$ *convex-cone T*
proof (*cases* $S=\{\} \vee T=\{\}$)
case *True*
then show *?thesis*
by (*auto dest*: *convex-cone-nonempty*)
next
case *False*
then have *convex-cone* ($S \times T$) $\Longrightarrow$ *convex-cone T*
by (*metis conic-Times-eq convex-cone-def convex-convex-hull convex-hull-Times hull-same times-eq-iff*)
then show *?thesis*
using *convex-cone-Times convex-cone-Times-D1* **by** *blast*
qed

lemma *convex-cone-hull-Un*:
convex-cone hull($S \cup T$) = ($\bigcup x \in$ *convex-cone hull S*. $\bigcup y \in$ *convex-cone hull T*. $\{x + y\}$)
(**is** *?lhs* = *?rhs*)
proof
show *?lhs* $\subseteq$ *?rhs*
proof (*rule hull-minimal*)
show $S \cup T \subseteq$ ($\bigcup x\in$*convex-cone hull S*. $\bigcup y\in$*convex-cone hull T*. $\{x + y\}$)
apply (*clarsimp simp*: *subset-iff*)
by (*metis add-0 convex-cone-hull-contains-0 group-cancel.rule0 hull-inc*)
show *convex-cone* ($\bigcup x\in$*convex-cone hull S*. $\bigcup y\in$*convex-cone hull T*. $\{x + y\}$)
by (*simp add*: *convex-cone-convex-cone-hull convex-cone-sums*)
qed
next
show *?rhs* $\subseteq$ *?lhs*
by *clarify* (*metis convex-cone-hull-add hull-mono le-sup-iff subsetD subsetI*)
qed

lemma *convex-cone-singleton* [*iff*]: *convex-cone* $\{0\}$
by (*simp add*: *convex-cone-iff*)

lemma *convex-hull-subset-convex-cone-hull*:
convex hull S $\subseteq$ *convex-cone hull S*
by (*simp add*: *convex-convex-cone-hull hull-minimal hull-subset*)

lemma *conic-hull-subset-convex-cone-hull*:
conic hull S $\subseteq$ *convex-cone hull S*
by (*simp add*: *conic-convex-cone-hull hull-minimal hull-subset*)

lemma *subspace-imp-convex-cone*: *subspace S* $\Longrightarrow$ *convex-cone S*
by (*simp add*: *convex-cone-iff subspace-def*)

lemma *convex-cone-span*: *convex-cone*(*span S*)
by (*simp add*: *subspace-imp-convex-cone*)

lemma *convex-cone-negations*:
convex-cone S $\Longrightarrow$ *convex-cone* (*image uminus S*)
by (*simp add*: *convex-cone-linear-image module-hom-uminus*)

lemma *subspace-convex-cone-symmetric*:
subspace S $\longleftrightarrow$ *convex-cone S* $\wedge$ ($\forall\, x \in S.\ -x \in S$)
by (*smt* (*verit*) *convex-cone-iff scaleR-left.minus subspace-def subspace-neg*)

lemma *convex-cone-hull-separate-nonempty*:
assumes $S \neq \{\}$
shows *convex-cone hull S* = *conic hull* (*convex hull S*) (**is** *?lhs* = *?rhs*)
proof
show *?lhs* $\subseteq$ *?rhs*
by (*simp add*: *assms conic-conic-hull conic-hull-eq-empty convex-cone-def convex-conic-hull hull-inc hull-minimal subsetI*)
show *?rhs* $\subseteq$ *?lhs*
by (*simp add*: *conic-convex-cone-hull convex-hull-subset-convex-cone-hull subset-hull*)
qed

lemma *convex-cone-hull-empty* [*simp*]: *convex-cone hull* {} = {*0*}
by (*metis convex-cone-hull-contains-0 convex-cone-singleton hull-redundant hull-same*)

lemma *convex-cone-hull-separate*:
convex-cone hull S = *insert 0* (*conic hull* (*convex hull S*))
by (*cases S*={}) (*simp-all add*: *convex-cone-hull-separate-nonempty insert-absorb*)

lemma *convex-cone-hull-convex-hull-nonempty*:
$S \neq \{\} \Longrightarrow$ *convex-cone hull S* = ($\bigcup x \in$ *convex hull S*. $\bigcup c\in\{0..\}.\ \{c *_R x\}$)
by (*force simp*: *convex-cone-hull-separate-nonempty conic-hull-as-image*)

lemma *convex-cone-hull-convex-hull*:
convex-cone hull S = *insert 0* ($\bigcup x \in$ *convex hull S*. $\bigcup c\in\{0..\}.\ \{c *_R x\}$)
by (*force simp*: *convex-cone-hull-separate conic-hull-as-image*)

lemma *convex-cone-hull-linear-image*:
linear f $\Longrightarrow$ *convex-cone hull* (f ‘ S) = *image f* (*convex-cone hull S*)
by (*metis* (*no-types*, *lifting*) *conic-hull-linear-image convex-cone-hull-separate convex-hull-linear-image image-insert linear-0*)

5.1 Finitely generated cone is polyhedral, and hence closed

proposition *polyhedron-convex-cone-hull*:
fixes S :: *'a::euclidean-space set*
assumes *finite S*
shows *polyhedron*(*convex-cone hull S*)
proof (*cases* $S = \{\}$)
case *True*
then show *?thesis*
by (*simp add*: *affine-imp-polyhedron*)
next
case *False*
then have *polyhedron*(*convex hull* (*insert 0 S*))
by (*simp add*: *assms polyhedron-convex-hull*)
then obtain *F a b* **where** *finite F*
and *F*: *convex hull* (*insert 0 S*) = $\bigcap$ *F*
and *ab*: $\bigwedge h.\ h \in F \Longrightarrow a\ h \neq 0 \wedge h = \{x.\ a\ h \cdot x \leq b\ h\}$
unfolding *polyhedron-def* **by** *metis*
then have $F \neq \{\}$
by (*metis bounded-convex-hull finite-imp-bounded Inf-empty assms finite-insert not-bounded-UNIV*)
show *?thesis*
unfolding *polyhedron-def*
proof (*intro exI conjI*)
show *convex-cone hull S* = $\bigcap\ \{h \in F.\ b\ h = 0\}$ (**is** *?lhs = ?rhs*)
proof
show *?lhs* $\subseteq$ *?rhs*
proof (*rule hull-minimal*)
show $S \subseteq \bigcap\ \{h \in F.\ b\ h = 0\}$
by (*smt* (*verit*, *best*) *F InterE InterI hull-subset insert-subset mem-Collect-eq subset-eq*)
have $\bigwedge S.\ [\![S \in F;\ b\ S = 0]\!] \Longrightarrow$ *convex-cone S*
by (*metis ab convex-cone-halfspace-le*)
then show *convex-cone* ($\bigcap\ \{h \in F.\ b\ h = 0\}$)
by (*force intro*: *convex-cone-Inter*)
qed
have $x \in$ *convex-cone hull S*
if x: $\bigwedge h.\ [\![h \in F;\ b\ h = 0]\!] \Longrightarrow x \in h$ **for** x
proof −
have $\exists t.\ 0 < t \wedge (t *_R x) \in h$ **if** $h \in F$ **for** h
proof (*cases* $b\ h = 0$)
case *True*
then show *?thesis*
by (*metis x linordered-field-no-ub mult-1 scaleR-one that zero-less-mult-iff*)

```
    next
      case False
      then have b h > 0
           by (smt (verit, del-insts) F InterE ab hull-subset inner-zero-right
insert-subset mem-Collect-eq that)
      then have 0 ∈ interior {x. a h · x ≤ b h}
        by (simp add: ab that)
      then have 0 ∈ interior h
        using ab that by auto
      then obtain ε where 0 < ε and ε: ball 0 ε ⊆ h
        using mem-interior by blast
      show ?thesis
      proof (cases x=0)
        case True
        then show ?thesis
          using ε ‹0 < ε› by auto
      next
        case False
        with ε ‹0 < ε› show ?thesis
          by (intro exI [where x=ε / (2 ∗ norm x)]) (auto simp: divide-simps)
      qed
    qed
    then obtain t where t: ⋀h. h ∈ F ⟹ 0 < t h ∧ (t h ∗R x) ∈ h
      by metis
    then have Inf (t ‘ F) ∗R x /R Inf (t ‘ F) = x
        by (smt (verit) ‹F ≠ {}› ‹finite F› field-simps(58) finite-imageI fi-
nite-less-Inf-iff image-iff image-is-empty)
    moreover have Inf (t ‘ F) ∗R x /R Inf (t ‘ F) ∈ convex-cone hull S
    proof (rule conicD [OF conic-convex-cone-hull])
      have Inf (t ‘ F) ∗R x ∈ ⋂ F
      proof clarify
        fix h
        assume h ∈ F
        have eq: Inf (t ‘ F) ∗R x = (1 − Inf(t ‘ F) / t h) ∗R 0 + (Inf(t ‘ F) /
t h) ∗R t h ∗R x
          using ‹h ∈ F› t by force
        show Inf (t ‘ F) ∗R x ∈ h
          unfolding eq
        proof (rule convexD-alt)
          have h = {x. a h · x ≤ b h}
            by (simp add: ‹h ∈ F› ab)
          then show convex h
            by (metis convex-halfspace-le)
          show 0 ∈ h
            by (metis F InterE ‹h ∈ F› hull-subset insertCI subsetD)
          show t h ∗R x ∈ h
            by (simp add: ‹h ∈ F› t)
          show 0 ≤ Inf (t ‘ F) / t h
                by (metis ‹F ≠ {}› ‹h ∈ F› cINF-greatest divide-nonneg-pos
```

less-eq-real-def t)
      show *Inf* (*t* ' *F*) / *t h* $\leq$ *1*
        by (*simp add*: ‹*finite F*› ‹*h* $\in$ *F*› *cInf-le-finite t*)
    qed
  qed
  moreover have *convex hull* (*insert 0 S*) $\subseteq$ *convex-cone hull S*
    by (*simp add*: *convex-cone-hull-contains-0 convex-convex-cone-hull hull-minimal hull-subset*)
  ultimately show *Inf* (*t* ' *F*) $*_R$ *x* $\in$ *convex-cone hull S*
    using *F* **by** *blast*
  show *0* $\leq$ *inverse* (*Inf* (*t* ' *F*))
  using *t* **by** (*simp add*: ‹*F* $\neq$ {}› ‹*finite F*› *finite-less-Inf-iff less-eq-real-def*)
 qed
 ultimately show *?thesis*
  by *auto*
qed
then show *?rhs* $\subseteq$ *?lhs*
 by *auto*
qed
show $\forall$ *h*$\in${*h* $\in$ *F*. *b h* = *0*}. $\exists$ *a b*. *a* $\neq$ *0* $\wedge$ *h* = {*x*. *a* $\cdot$ *x* $\leq$ *b*}
 using *ab* **by** *blast*
qed (*auto simp*: ‹*finite F*›)
qed

lemma *closed-convex-cone-hull*:
 fixes *S* :: *'a*::*euclidean-space set*
 shows *finite S* $\Longrightarrow$ *closed*(*convex-cone hull S*)
 by (*simp add*: *polyhedron-convex-cone-hull polyhedron-imp-closed*)

lemma *polyhedron-convex-cone-hull-polytope*:
 fixes *S* :: *'a*::*euclidean-space set*
 shows *polytope S* $\Longrightarrow$ *polyhedron*(*convex-cone hull S*)
 by (*metis convex-cone-hull-separate hull-hull polyhedron-convex-cone-hull polytope-def*)

lemma *polyhedron-conic-hull-polytope*:
 fixes *S* :: *'a*::*euclidean-space set*
 shows *polytope S* $\Longrightarrow$ *polyhedron*(*conic hull S*)
 by (*metis conic-hull-eq-empty convex-cone-hull-separate-nonempty hull-hull polyhedron-convex-cone-hull-polytope polyhedron-empty polytope-def*)

lemma *closed-conic-hull-strong*:
 fixes *S* :: *'a*::*euclidean-space set*
 shows *0* $\in$ *rel-interior S* $\vee$ *polytope S* $\vee$ *compact S* $\wedge$ ~(*0* $\in$ *S*) $\Longrightarrow$ *closed*(*conic hull S*)
 using *closed-conic-hull polyhedron-conic-hull-polytope polyhedron-imp-closed* **by** *blast*

end

6 Inclusion-exclusion principle

Inclusion-exclusion principle, the usual and generalized forms.

theory *Inclusion-Exclusion*
  imports *Main*
begin

lemma *subset-insert-lemma*:
  $\{T.\ T \subseteq (\mathit{insert}\ a\ S) \wedge P\ T\} = \{T.\ T \subseteq S \wedge P\ T\} \cup \{\mathit{insert}\ a\ T\ |T.\ T \subseteq S \wedge P(\mathit{insert}\ a\ T)\}$ (**is** *?L=?R*)
proof
  show $?L \subseteq ?R$
    by (*smt* (*verit*) *UnI1 UnI2 insert-Diff mem-Collect-eq subsetI subset-insert-iff*)
qed *blast*

locale *Incl-Excl* =
  fixes $P :: 'a\ \mathit{set} \Rightarrow \mathit{bool}$ **and** $f :: 'a\ \mathit{set} \Rightarrow 'b::\mathit{ring\text{-}1}$
  assumes *disj-add*: $[\![P\ S;\ P\ T;\ \mathit{disjnt}\ S\ T]\!] \Longrightarrow f(S \cup T) = f\ S + f\ T$
    and *empty*: $P\{\}$
    and *Int*: $[\![P\ S;\ P\ T]\!] \Longrightarrow P(S \cap T)$
    and *Un*: $[\![P\ S;\ P\ T]\!] \Longrightarrow P(S \cup T)$
    and *Diff*: $[\![P\ S;\ P\ T]\!] \Longrightarrow P(S - T)$

begin

lemma *f-empty* [*simp*]: $f\{\} = 0$
  using *disj-add empty* **by** *fastforce*

lemma *f-Un-Int*: $[\![P\ S;\ P\ T]\!] \Longrightarrow f(S \cup T) + f(S \cap T) = f\ S + f\ T$
  by (*smt* (*verit*, *ccfv-threshold*) *Groups.add-ac*(*2*) *Incl-Excl.Diff Incl-Excl.Int Incl-Excl-axioms Int-Diff-Un Int-Diff-disjoint Int-absorb Un-Diff Un-Int-eq*(*2*) *disj-add disjnt-def group-cancel.add2 sup-bot.right-neutral*)

lemma *restricted-indexed*:
  assumes *finite A* **and** X: $\bigwedge a.\ a \in A \Longrightarrow P(X\ a)$
  shows $f(\bigcup(X\ `\ A)) = (\sum B\ |\ B \subseteq A \wedge B \neq \{\}.\ (-\ 1)\ \hat{}\ (\mathit{card}\ B + 1) * f\ (\bigcap (X\ `\ B)))$
proof −
  have $[\![\mathit{finite}\ A;\ \mathit{card}\ A = n;\ \forall a \in A.\ P\ (X\ a)]\!]$
      $\Longrightarrow f(\bigcup(X\ `\ A)) = (\sum B\ |\ B \subseteq A \wedge B \neq \{\}.\ (-\ 1)\ \hat{}\ (\mathit{card}\ B + 1) * f\ (\bigcap\ (X\ `\ B)))$ **for** n X **and** $A :: 'c\ \mathit{set}$
  proof (*induction n arbitrary*: *A X rule*: *less-induct*)
    case (*less n0 A0 X*)
    show *?case*
    proof (*cases n0=0*)
      case *True*

with *less* **show** *?thesis*
by *fastforce*
next
case *False*
with *less.prems* **obtain** $A\ n\ a$ **where** $*$: $n0 = Suc\ n\ A0 = insert\ a\ A\ a \notin A\ card\ A = n\ finite\ A$
by (*metis card-Suc-eq-finite not0-implies-Suc*)
with *less* **have** $P\ (X\ a)$ **by** *blast*
have *APX*: $\forall a \in A.\ P\ (X\ a)$
by (*simp add*: $*$ *less.prems*)
have *PUXA*: $P\ (\bigcup\ (X\ `\ A))$
using ‹*finite A*› *APX*
by (*induction*) (*auto simp*: *empty Un*)
have $f\ (\bigcup\ (X\ `\ A0)) = f\ (X\ a \cup \bigcup\ (X\ `\ A))$
by (*simp add*: $*$)
also have ... $= f\ (X\ a) + f\ (\bigcup\ (X\ `\ A)) - f\ (X\ a \cap \bigcup\ (X\ `\ A))$
using *f-Un-Int add-diff-cancel PUXA* ‹$P\ (X\ a)$› **by** *metis*
also have ... $= f\ (X\ a) - (\sum B \mid B \subseteq A \wedge B \neq \{\}.\ (-\ 1)\ \hat{}\ card\ B * f\ (\bigcap\ (X\ `\ B))) +$
$(\sum B \mid B \subseteq A \wedge B \neq \{\}.\ (-\ 1)\ \hat{}\ card\ B * f\ (X\ a \cap \bigcap\ (X\ `\ B)))$
proof $-$
have *1*: $f\ (\bigcup i{\in}A.\ X\ a \cap X\ i) = (\sum B \mid B \subseteq A \wedge B \neq \{\}.\ (-\ 1)\ \hat{}\ (card\ B + 1) * f\ (\bigcap b{\in}B.\ X\ a \cap X\ b))$
using *less.IH* [*of n A* $\lambda i.\ X\ a \cap X\ i$] *APX Int* ‹$P\ (X\ a)$› **by** (*simp add*: $*$)
have *2*: $X\ a \cap \bigcup\ (X\ `\ A) = (\bigcup i{\in}A.\ X\ a \cap X\ i)$
by *auto*
have *3*: $f\ (\bigcup\ (X\ `\ A)) = (\sum B \mid B \subseteq A \wedge B \neq \{\}.\ (-\ 1)\ \hat{}\ (card\ B + 1) * f\ (\bigcap\ (X\ `\ B)))$
using *less.IH* [*of n A X*] *APX Int* ‹$P\ (X\ a)$› **by** (*simp add*: $*$)
show *?thesis*
unfolding *3 2 1*
by (*simp add*: *sum-negf*)
qed
also have ... $= (\sum B \mid B \subseteq A0 \wedge B \neq \{\}.\ (-\ 1)\ \hat{}\ (card\ B + 1) * f\ (\bigcap\ (X\ `\ B)))$
proof $-$
have *F*: $\{insert\ a\ B \mid B.\ B \subseteq A\} = insert\ a\ `\ Pow\ A \wedge \{B.\ B \subseteq A \wedge B \neq \{\}\} = Pow\ A - \{\{\}\}$
by *auto*
have *G*: $(\sum B{\in}Pow\ A.\ (-\ 1)\ \hat{}\ card\ (insert\ a\ B) * f\ (X\ a \cap \bigcap\ (X\ `\ B))) = (\sum B{\in}Pow\ A.\ -\ ((-\ 1)\ \hat{}\ card\ B * f\ (X\ a \cap \bigcap\ (X\ `\ B))))$
proof (*rule sum.cong* [*OF refl*])
fix B
assume B: $B \in Pow\ A$
then have *finite B*
using ‹*finite A*› *finite-subset* **by** *auto*
show $(-\ 1)\ \hat{}\ card\ (insert\ a\ B) * f\ (X\ a \cap \bigcap\ (X\ `\ B)) = -\ ((-\ 1)\ \hat{}\ card\ B * f\ (X\ a \cap \bigcap\ (X\ `\ B)))$
using $B\ *$ **by** (*auto simp add*: *card-insert-if* ‹*finite B*›)

qed
have *disj*: $\{B.\ B \subseteq A \wedge B \neq \{\}\} \cap \{insert\ a\ B\ |B.\ B \subseteq A\} = \{\}$
using $*$ **by** *blast*
have *inj*: *inj-on* (*insert a*) (*Pow A*)
using $*$ *inj-on-def* **by** *fastforce*
show *?thesis*
apply (*simp add*: $*$ *subset-insert-lemma sum.union-disjoint disj sum-negf*)
apply (*simp add*: *F G sum-negf sum.reindex* [*OF inj*] *o-def sum-diff* $*$)
done
qed
finally show *?thesis* **.**
qed
qed
then show *?thesis*
by (*meson assms*)
qed

lemma *restricted*:
assumes *finite A* $\bigwedge a.\ a \in A \Longrightarrow P\ a$
shows $f(\bigcup\ A) = (\sum B\ |\ B \subseteq A \wedge B \neq \{\}.\ (-\ 1)\ \hat{}\ (card\ B + 1) * f\ (\bigcap\ B))$
using *restricted-indexed* [*of A* $\lambda x.\ x$] *assms* **by** *auto*

end

6.1 Versions for unrestrictedly additive functions

lemma *Incl-Excl-UN*:
fixes $f :: {'}a\ set \Rightarrow {'}b{::}ring\text{-}1$
assumes $\bigwedge S\ T.\ disjnt\ S\ T \Longrightarrow f(S \cup T) = f\ S + f\ T$ *finite A*
shows $f(\bigcup(G\ `\ A)) = (\sum B\ |\ B \subseteq A \wedge B \neq \{\}.\ (-1)\ \hat{}\ (card\ B + 1) * f\ (\bigcap (G\ `\ B)))$
proof –
interpret *Incl-Excl* $\lambda x.$ *True f*
by (*simp add*: *Incl-Excl.intro assms*(*1*))
show *?thesis*
using *restricted-indexed assms* **by** *blast*
qed

lemma *Incl-Excl-Union*:
fixes $f :: {'}a\ set \Rightarrow {'}b{::}ring\text{-}1$
assumes $\bigwedge S\ T.\ disjnt\ S\ T \Longrightarrow f(S \cup T) = f\ S + f\ T$ *finite A*
shows $f(\bigcup\ A) = (\sum B\ |\ B \subseteq A \wedge B \neq \{\}.\ (-\ 1)\ \hat{}\ (card\ B + 1) * f\ (\bigcap\ B))$
using *Incl-Excl-UN*[*of f A* $\lambda X.\ X$] *assms* **by** *simp*

The famous inclusion-exclusion formula for the cardinality of a union

lemma *int-card-UNION*:
assumes *finite A* $\bigwedge K.\ K \in A \Longrightarrow$ *finite K*
shows $int\ (card\ (\bigcup A)) = (\sum I\ |\ I \subseteq A \wedge I \neq \{\}.\ (-\ 1)\ \hat{}\ (card\ I + 1) * int\ (card\ (\bigcap I)))$
proof –

```
  interpret Incl-Excl finite int o card
  proof qed (auto simp add: card-Un-disjnt)
  show ?thesis
    using restricted assms by auto
qed
```

A more conventional form

```
lemma inclusion-exclusion:
  assumes finite A ⋀K. K ∈ A ⟹ finite K
  shows int(card(⋃ A)) =
        (∑ n=1..card A. (−1) ^(Suc n) ∗ (∑ B | B ⊆ A ∧ card B = n. int (card
(⋂ B)))) (is -=?R)
proof −
  have fin: finite {I. I ⊆ A ∧ I ≠ {}}
    by (simp add: assms)
  have ⋀k. ⟦Suc 0 ≤ k; k ≤ card A⟧ ⟹ ∃ B⊆A. B ≠ {} ∧ k = card B
   by (metis (mono-tags, lifting) Suc-le-D Zero-neq-Suc card-eq-0-iff obtain-subset-with-card-n)
  with ‹finite A› finite-subset
  have card-eq: card ‘ {I. I ⊆ A ∧ I ≠ {}} = {1..card A}
    using not-less-eq-eq card-mono by (fastforce simp: image-iff)
  have int(card(⋃ A))
      = (∑ y = 1..card A. ∑ I∈{x. x ⊆ A ∧ x ≠ {} ∧ card x = y}. − ((− 1) ^ y
∗ int (card (⋂ I))))
    by (simp add: int-card-UNION assms sum.image-gen [OF fin, where g=card]
card-eq)
  also have ... = ?R
  proof −
    have {B. B ⊆ A ∧ B ≠ {} ∧ card B = k} = {B. B ⊆ A ∧ card B = k}
      if Suc 0 ≤ k and k ≤ card A for k
      using that by auto
    then show ?thesis
      by (clarsimp simp add: sum-negf simp flip: sum-distrib-left)
  qed
  finally show ?thesis .
qed

lemma card-UNION:
  assumes finite A and ⋀K. K ∈ A ⟹ finite K
  shows card (⋃A) = nat (∑I | I ⊆ A ∧ I ≠ {}. (− 1) ^(card I + 1) ∗ int
(card (⋂I)))
  by (simp only: flip: int-card-UNION [OF assms])

lemma card-UNION-nonneg:
  assumes finite A and ⋀K. K ∈ A ⟹ finite K
  shows (∑I | I ⊆ A ∧ I ≠ {}. (− 1) ^(card I + 1) ∗ int (card (⋂I))) ≥ 0
  using int-card-UNION [OF assms] by presburger
```

6.2 a general "Moebius inversion" inclusion-exclusion principle. This "symmetric" form is from Ira Gessel: "Symmetric Inclusion-Exclusion"

lemma *sum-Un-eq*:
$[\![S \cap T = \{\}; S \cup T = U; \textit{finite } U]\!]$
$\Longrightarrow (\textit{sum } f\ S + \textit{sum } f\ T = \textit{sum } f\ U)$
by (*metis finite-Un sum.union-disjoint*)

lemma *card-adjust-lemma*: $[\![\textit{inj-on } f\ S;\ x = y + \textit{card } (f \text{ ‘ } S)]\!] \Longrightarrow x = y + \textit{card } S$
by (*simp add*: *card-image*)

lemma *card-subsets-step*:
assumes $\textit{finite } S\ x \notin S\ U \subseteq S$
shows $\textit{card } \{T.\ T \subseteq (\textit{insert } x\ S) \wedge U \subseteq T \wedge \textit{odd}(\textit{card } T)\}$
$= \textit{card } \{T.\ T \subseteq S \wedge U \subseteq T \wedge \textit{odd}(\textit{card } T)\} + \textit{card } \{T.\ T \subseteq S \wedge U \subseteq T$
$\wedge\ \textit{even}(\textit{card } T)\} \wedge$
$\textit{card } \{T.\ T \subseteq (\textit{insert } x\ S) \wedge U \subseteq T \wedge \textit{even}(\textit{card } T)\}$
$= \textit{card } \{T.\ T \subseteq S \wedge U \subseteq T \wedge \textit{even}(\textit{card } T)\} + \textit{card } \{T.\ T \subseteq S \wedge U \subseteq T$
$\wedge\ \textit{odd}(\textit{card } T)\}$
proof −
have *inj*: $\textit{inj-on } (\textit{insert } x)\ \{T.\ T \subseteq S \wedge P\ T\}$ **for** P
using *assms* **by** (*auto simp*: *inj-on-def*)
have [*simp*]: $\textit{finite } \{T.\ T \subseteq S \wedge P\ T\}\ \ \textit{finite } (\textit{insert } x \text{ ‘ } \{T.\ T \subseteq S \wedge P\ T\})$
for P
using ‹*finite S*› **by** *auto*
have [*simp*]: $\textit{disjnt } \{T.\ T \subseteq S \wedge P\ T\}\ (\textit{insert } x \text{ ‘ } \{T.\ T \subseteq S \wedge Q\ T\})$ **for** $P\ Q$
using *assms* **by** (*auto simp*: *disjnt-iff*)
have *eq*: $\{T.\ T \subseteq S \wedge U \subseteq T \wedge P\ T\} \cup \textit{insert } x \text{ ‘ } \{T.\ T \subseteq S \wedge U \subseteq T \wedge Q$
$T\}$
$= \{T.\ T \subseteq \textit{insert } x\ S \wedge U \subseteq T \wedge P\ T\}$ (**is** $?L = ?R$)
if $\bigwedge A.\ A \subseteq S \Longrightarrow Q\ (\textit{insert } x\ A) \longleftrightarrow P\ A\ \bigwedge A.\ \neg\ Q\ A \longleftrightarrow P\ A$ **for** $P\ Q$
proof
show $?L \subseteq ?R$
by (*clarsimp simp*: *image-iff subset-iff*) (*meson subsetI that*)
show $?R \subseteq ?L$
using ‹$U \subseteq S$›
by (*clarsimp simp*: *image-iff*) (*smt* (*verit*) *insert-iff mk-disjoint-insert subset-iff*
that)
qed
have [*simp*]: $\bigwedge A.\ A \subseteq S \Longrightarrow \textit{even } (\textit{card } (\textit{insert } x\ A)) \longleftrightarrow \textit{odd } (\textit{card } A)$
by (*metis* ‹*finite S*› ‹$x \notin S$› *card-insert-disjoint even-Suc finite-subset subsetD*)
show *?thesis*
by (*intro conjI card-adjust-lemma* [*OF inj*]; *simp add*: *eq flip*: *card-Un-disjnt*)
qed

lemma *card-subsupersets-even-odd*:
assumes $\textit{finite } S\ U \subset S$
shows $\textit{card } \{T.\ T \subseteq S \wedge U \subseteq T \wedge \textit{even}(\textit{card } T)\}$

$= card\ \{T.\ T \subseteq S \wedge U \subseteq T \wedge odd(card\ T)\}$
using *assms*
proof (*induction card S arbitrary*: *S rule*: *less-induct*)
case (*less S*)
then obtain x **where** $x \notin U\ x \in S$
by *blast*
then have U: $U \subseteq S - \{x\}$
using *less.prems*(*2*) **by** *blast*
let $?V = S - \{x\}$
show *?case*
using *card-subsets-step* [*of ?V x U*] *less.prems U*
by (*simp add*: *insert-absorb* ‹$x \in S$›)
qed

lemma *sum-alternating-cancels*:
assumes *finite S* $card\ \{x.\ x \in S \wedge even(f\ x)\} = card\ \{x.\ x \in S \wedge odd(f\ x)\}$
shows $(\sum x{\in}S.\ (-1)\ \hat{}\ f\ x) = (0{::}'b{::}ring\text{-}1)$
proof −
have $(\sum x{\in}S.\ (-1)\ \hat{}\ f\ x)$
$= (\sum x \mid x \in S \wedge even\ (f\ x).\ (-1)\ \hat{}\ f\ x) + (\sum x \mid x \in S \wedge odd\ (f\ x).\ (-1)$
$\hat{}\ f\ x)$
by (*rule sum-Un-eq* [*symmetric*]; *force simp*: ‹*finite S*›)
also have ... $= (0{::}'b{::}ring\text{-}1)$
by (*simp add*: *minus-one-power-iff assms cong*: *conj-cong*)
finally show *?thesis* **.**
qed

lemma *inclusion-exclusion-symmetric*:
fixes $f :: 'a\ set \Rightarrow 'b{::}ring\text{-}1$
assumes §: $\bigwedge S.\ finite\ S \Longrightarrow g\ S = (\sum T \in Pow\ S.\ (-1)\ \hat{}\ card\ T * f\ T)$
and *finite S*
shows $f\ S = (\sum T \in Pow\ S.\ (-1)\ \hat{}\ card\ T * g\ T)$
proof −
have $(-1)\ \hat{}\ card\ T * g\ T = (-1)\ \hat{}\ card\ T * (\sum U \mid U \subseteq S \wedge U \subseteq T.\ (-1)$
$\hat{}\ card\ U * f\ U)$
if $T \subseteq S$ **for** T
proof −
have [*simp*]: $\{U.\ U \subseteq S \wedge U \subseteq T\} = Pow\ T$
using *that* **by** *auto*
show *?thesis*
using *that* **by** (*simp add*: ‹*finite S*› *finite-subset* §)
qed
then have $(\sum T \in Pow\ S.\ (-1)\ \hat{}\ card\ T * g\ T)$
$= (\sum T{\in}Pow\ S.\ (-1)\ \hat{}\ card\ T * (\sum U \mid U \in \{U.\ U \subseteq S\} \wedge U \subseteq T.\ (-1)$
$\hat{}\ card\ U * f\ U))$
by *simp*
also have ... $= (\sum U{\in}Pow\ S.\ (\sum T \mid T \subseteq S \wedge U \subseteq T.\ (-1)\ \hat{}\ card\ T) * (-1)$
$\hat{}\ card\ U * f\ U)$
unfolding *sum-distrib-left*

```
    by (subst sum.swap-restrict; simp add: ‹finite S› algebra-simps sum-distrib-right
Pow-def)
  also have ... = (∑ U∈Pow S. if U=S then f S else 0)
  proof −
    have [simp]: {T. T ⊆ S ∧ S ⊆ T} = {S}
      by auto
    show ?thesis
      apply (rule sum.cong [OF refl])
       by (simp add: sum-alternating-cancels card-subsupersets-even-odd ‹finite S›
flip: power-add)
  qed
  also have ... = f S
    by (simp add: ‹finite S›)
  finally show ?thesis
    by presburger
qed
```

The more typical non-symmetric version.

```
lemma inclusion-exclusion-mobius:
  fixes f :: 'a set ⇒ 'b::ring-1
  assumes §: ⋀S. finite S ⟹ g S = sum f (Pow S) and finite S
  shows f S = (∑ T ∈ Pow S. (−1) ^(card S − card T) ∗ g T) (is - = ?rhs)
proof −
  have (− 1) ^ card S ∗ f S = (∑ T∈Pow S. (− 1) ^ card T ∗ g T)
   by (rule inclusion-exclusion-symmetric; simp add: assms flip: power-add mult.assoc)
   then have ((− 1) ^ card S ∗ (− 1) ^ card S) ∗ f S = ((− 1) ^ card S) ∗
(∑ T∈Pow S. (− 1) ^ card T ∗ g T)
    by (simp add: mult-ac)
  then have f S = (∑ T∈Pow S. (− 1) ^(card S + card T) ∗ g T)
    by (simp add: sum-distrib-left flip: power-add mult.assoc)
  also have ... = ?rhs
    by (simp add: ‹finite S› card-mono neg-one-power-add-eq-neg-one-power-diff)
  finally show ?thesis .
qed

end
```

7 Euler's Polyhedron Formula

One of the Famous 100 Theorems, ported from HOL Light

Cited source: Lawrence, J. (1997). A Short Proof of Euler's Relation for Convex Polytopes. *Canadian Mathematical Bulletin*, **40**(4), 471–474.

```
theory Euler-Formula
  imports
    HOL−Analysis.Analysis
    Library-Extras
    Inclusion-Exclusion
begin
```

Interpret which "side" of a hyperplane a point is on.

definition *hyperplane-side*
where $hyperplane\text{-}side \equiv \lambda(a,b).\ \lambda x.\ sgn\ (a \cdot x - b)$

Equivalence relation imposed by a hyperplane arrangement.

definition *hyperplane-equiv*
where $hyperplane\text{-}equiv \equiv \lambda A\ x\ y.\ \forall h \in A.\ hyperplane\text{-}side\ h\ x = hyperplane\text{-}side\ h\ y$

lemma *hyperplane-equiv-refl* [*iff*]: *hyperplane-equiv A x x*
by (*simp add*: *hyperplane-equiv-def*)

lemma *hyperplane-equiv-sym*:
$hyperplane\text{-}equiv\ A\ x\ y \longleftrightarrow hyperplane\text{-}equiv\ A\ y\ x$
by (*auto simp*: *hyperplane-equiv-def*)

lemma *hyperplane-equiv-trans*:
$[\![hyperplane\text{-}equiv\ A\ x\ y;\ hyperplane\text{-}equiv\ A\ y\ z]\!] \Longrightarrow hyperplane\text{-}equiv\ A\ x\ z$
by (*auto simp*: *hyperplane-equiv-def*)

lemma *hyperplane-equiv-Un*:
$hyperplane\text{-}equiv\ (A \cup B)\ x\ y \longleftrightarrow hyperplane\text{-}equiv\ A\ x\ y \wedge hyperplane\text{-}equiv\ B\ x\ y$
by (*meson Un-iff hyperplane-equiv-def*)

7.1 Cells of a hyperplane arrangement

definition $hyperplane\text{-}cell :: ('a{::}real\text{-}inner \times real)\ set \Rightarrow 'a\ set \Rightarrow bool$
where $hyperplane\text{-}cell \equiv \lambda A\ C.\ \exists x.\ C = Collect\ (hyperplane\text{-}equiv\ A\ x)$

lemma *hyperplane-cell*: $hyperplane\text{-}cell\ A\ C \longleftrightarrow (\exists x.\ C = \{y.\ hyperplane\text{-}equiv\ A\ x\ y\})$
by (*simp add*: *hyperplane-cell-def*)

lemma *not-hyperplane-cell-empty* [*simp*]: $\neg\ hyperplane\text{-}cell\ A\ \{\}$
using *hyperplane-cell* **by** *auto*

lemma *nonempty-hyperplane-cell*: $hyperplane\text{-}cell\ A\ C \Longrightarrow (C \neq \{\})$
by *auto*

lemma *Union-hyperplane-cells*: $\bigcup\ \{C.\ hyperplane\text{-}cell\ A\ C\} = UNIV$
using *hyperplane-cell* **by** *blast*

lemma *disjoint-hyperplane-cells*:
$[\![hyperplane\text{-}cell\ A\ C1;\ hyperplane\text{-}cell\ A\ C2;\ C1 \neq C2]\!] \Longrightarrow disjnt\ C1\ C2$
by (*force simp*: *hyperplane-cell-def disjnt-iff hyperplane-equiv-def*)

lemma *disjoint-hyperplane-cells-eq*:

$\llbracket$*hyperplane-cell A C1*; *hyperplane-cell A C2*$\rrbracket \Longrightarrow$ (*disjnt C1 C2* $\longleftrightarrow$ (*C1* $\neq$ *C2*))
using *disjoint-hyperplane-cells* **by** *auto*

lemma *hyperplane-cell-empty* [*iff*]: *hyperplane-cell* {} *C* $\longleftrightarrow$ *C* = *UNIV*
by (*simp add*: *hyperplane-cell hyperplane-equiv-def*)

lemma *hyperplane-cell-singleton-cases*:
assumes *hyperplane-cell* {(*a*,*b*)} *C*
shows *C* = {*x*. *a* $\cdot$ *x* = *b*} $\vee$ *C* = {*x*. *a* $\cdot$ *x* < *b*} $\vee$ *C* = {*x*. *a* $\cdot$ *x* > *b*}
proof −
obtain *x* **where** *x*: *C* = {*y*. *hyperplane-side* (*a*, *b*) *x* = *hyperplane-side* (*a*, *b*) *y*}
using *assms* **by** (*auto simp*: *hyperplane-equiv-def hyperplane-cell*)
then show *?thesis*
by (*auto simp*: *hyperplane-side-def sgn-if split*: *if-split-asm*)
qed

lemma *hyperplane-cell-singleton*:
hyperplane-cell {(*a*,*b*)} *C* $\longleftrightarrow$
(*if a* = *0 then C* = *UNIV else C* = {*x*. *a* $\cdot$ *x* = *b*} $\vee$ *C* = {*x*. *a* $\cdot$ *x* < *b*} $\vee$ *C* = {*x*. *a* $\cdot$ *x* > *b*})
apply (*simp add*: *hyperplane-cell-def hyperplane-equiv-def hyperplane-side-def sgn-if split*: *if-split-asm*)
by (*smt* (*verit*) *Collect-cong gt-ex hyperplane-eq-Ex lt-ex*)

lemma *hyperplane-cell-Un*:
hyperplane-cell (*A* $\cup$ *B*) *C* $\longleftrightarrow$
C $\neq$ {} $\wedge$
($\exists$ *C1 C2*. *hyperplane-cell A C1* $\wedge$ *hyperplane-cell B C2* $\wedge$ *C* = *C1* $\cap$ *C2*)
by (*auto simp*: *hyperplane-cell hyperplane-equiv-def*)

lemma *finite-hyperplane-cells*:
finite A $\Longrightarrow$ *finite* {*C*. *hyperplane-cell A C*}
proof (*induction rule*: *finite-induct*)
case (*insert p A*)
obtain *a b* **where** *peq*: *p* = (*a*,*b*)
by *fastforce*
have *Collect* (*hyperplane-cell* {*p*}) $\subseteq$ {{*x*. *a* $\cdot$ *x* = *b*},{*x*. *a* $\cdot$ *x* < *b*},{*x*. *a* $\cdot$ *x* > *b*}}
using *hyperplane-cell-singleton-cases*
by (*auto simp*: *peq*)
then have $*$: *finite* (*Collect* (*hyperplane-cell* {*p*}))
by (*simp add*: *finite-subset*)
define $\mathcal{C}$ **where** $\mathcal{C} \equiv (\bigcup$ *C1* $\in$ {*C*. *hyperplane-cell A C*}. $\bigcup$ *C2* $\in$ {*C*. *hyperplane-cell* {*p*} *C*}. {*C1* $\cap$ *C2*})
have {*a*. *hyperplane-cell* (*insert p A*) *a*} $\subseteq \mathcal{C}$
using *hyperplane-cell-Un* [*of* {*p*} *A*] **by** (*auto simp*: $\mathcal{C}$*-def*)
moreover have *finite* $\mathcal{C}$
using $*$ $\mathcal{C}$*-def insert.IH* **by** *blast*

ultimately show *?case*
using *finite-subset* **by** *blast*
qed *auto*

lemma *finite-restrict-hyperplane-cells*:
finite $A \Longrightarrow$ *finite* $\{C.$ *hyperplane-cell* $A\ C \wedge P\ C\}$
by (*simp add*: *finite-hyperplane-cells*)

lemma *finite-set-of-hyperplane-cells*:
$[\![$*finite* A; $\bigwedge C.\ C \in \mathcal{C} \Longrightarrow$ *hyperplane-cell* $A\ C]\!] \Longrightarrow$ *finite* $\mathcal{C}$
by (*metis finite-hyperplane-cells finite-subset mem-Collect-eq subsetI*)

lemma *pairwise-disjoint-hyperplane-cells*:
$(\bigwedge C.\ C \in \mathcal{C} \Longrightarrow$ *hyperplane-cell* $A\ C) \Longrightarrow$ *pairwise disjnt* $\mathcal{C}$
by (*metis disjoint-hyperplane-cells pairwiseI*)

lemma *hyperplane-cell-Int-open-affine*:
assumes *finite* A *hyperplane-cell* $A\ C$
obtains $S\ T$ **where** *open* S *affine* $T\ C = S \cap T$
using *assms*
proof (*induction arbitrary*: *thesis* C *rule*: *finite-induct*)
case *empty*
then show *?case*
by *auto*
next
case (*insert* $p\ A$ *thesis* C')
obtain $a\ b$ **where** *peq*: $p = (a,b)$
by *fastforce*
obtain $C\ C1$ **where** $C1$: *hyperplane-cell* $\{(a,b)\}\ C1$ **and** C: *hyperplane-cell* A
C
and $C' \neq \{\}$ **and** C': $C' = C1 \cap C$
by (*metis hyperplane-cell-Un insert.prems*(*2*) *insert-is-Un peq*)
then obtain $S\ T$ **where** ST: *open* S *affine* $T\ C = S \cap T$
by (*meson insert.IH*)
show *?case*
proof (*cases* $a=0$)
case *True*
with *insert.prems* **show** *?thesis*
by (*metis C1 Int-commute ST* ‹$C' = C1 \cap C$› *hyperplane-cell-singleton*
inf-top.right-neutral)
next
case *False*
then consider $C1 = \{x.\ a \cdot x = b\} \mid C1 = \{x.\ a \cdot x < b\} \mid C1 = \{x.\ b < a$
$\cdot\ x\}$
by (*metis C1 hyperplane-cell-singleton*)
then show *?thesis*
proof *cases*
case *1*
then show *thesis*

```
      by (metis C′ ST affine-Int affine-hyperplane inf-left-commute insert.prems(1))
    next
      case 2
      with ST show thesis
          by (metis Int-assoc C′ insert.prems(1) open-Int open-halfspace-lt)
    next
      case 3
      with ST show thesis
        by (metis Int-assoc C′ insert.prems(1) open-Int open-halfspace-gt)
    qed
  qed
qed

lemma hyperplane-cell-relatively-open:
  assumes finite A hyperplane-cell A C
  shows openin (subtopology euclidean (affine hull C)) C
proof −
  obtain S T where open S affine T C = S ∩ T
    by (meson assms hyperplane-cell-Int-open-affine)
  show ?thesis
  proof (cases S ∩ T = {})
    case True
    then show ?thesis
      by (simp add: ‹C = S ∩ T›)
  next
    case False
    then have affine hull (S ∩ T) = T
     by (metis ‹affine T› ‹open S› affine-hull-affine-Int-open hull-same inf-commute)
    then show ?thesis
      using ‹C = S ∩ T› ‹open S› openin-subtopology by fastforce
  qed
qed

lemma hyperplane-cell-relative-interior:
   ⟦finite A; hyperplane-cell A C⟧ =⇒ rel-interior C = C
  by (simp add: hyperplane-cell-relatively-open rel-interior-openin)

lemma hyperplane-cell-convex:
  assumes hyperplane-cell A C
  shows convex C
proof −
  obtain c where c: C = {y. hyperplane-equiv A c y}
    by (meson assms hyperplane-cell)
  have convex (⋂h∈A. {y. hyperplane-side h c = hyperplane-side h y})
  proof (rule convex-INT)
    fix h :: ′a × real
    assume h ∈ A
    obtain a b where heq: h = (a,b)
      by fastforce
```

have [*simp*]: $\{y.\ \neg\ a \cdot c < a \cdot y \wedge a \cdot y = a \cdot c\} = \{y.\ a \cdot y = a \cdot c\}$
$\{y.\ \neg\ b < a \cdot y \wedge a \cdot y \neq b\} = \{y.\ b > a \cdot y\}$
by *auto*
then show *convex* $\{y.$ *hyperplane-side h c* $=$ *hyperplane-side h y*$\}$
by (*fastforce simp*: *heq hyperplane-side-def sgn-if convex-halfspace-gt convex-halfspace-lt convex-hyperplane cong*: *conj-cong*)
qed
with *c* **show** *?thesis*
by (*simp add*: *hyperplane-equiv-def INTER-eq*)
qed

lemma *hyperplane-cell-Inter*:
assumes $\bigwedge C.\ C \in \mathcal{C} \Longrightarrow$ *hyperplane-cell A C*
and $\mathcal{C} \neq \{\}$ **and** *INT*: $\bigcap \mathcal{C} \neq \{\}$
shows *hyperplane-cell A* $(\bigcap \mathcal{C})$
proof −
have $\bigcap \mathcal{C} = \{y.$ *hyperplane-equiv A z y*$\}$
if $z \in \bigcap \mathcal{C}$ **for** *z*
using *assms that* **by** (*force simp*: *hyperplane-cell hyperplane-equiv-def*)
with *INT hyperplane-cell* **show** *?thesis*
by *fastforce*
qed

lemma *hyperplane-cell-Int*:
$[\![$*hyperplane-cell A S*; *hyperplane-cell A T*; $S \cap T \neq \{\}]\!] \Longrightarrow$ *hyperplane-cell A* $(S \cap T)$
by (*metis hyperplane-cell-Un sup.idem*)

7.2 A cell complex is considered to be a union of such cells

definition *hyperplane-cellcomplex*
where *hyperplane-cellcomplex A S* $\equiv$
$\exists \mathcal{T}.\ (\forall C \in \mathcal{T}.$ *hyperplane-cell A C*$) \wedge S = \bigcup \mathcal{T}$

lemma *hyperplane-cellcomplex-empty* [*simp*]: *hyperplane-cellcomplex A* $\{\}$
using *hyperplane-cellcomplex-def* **by** *auto*

lemma *hyperplane-cell-cellcomplex*:
hyperplane-cell A C $\Longrightarrow$ *hyperplane-cellcomplex A C*
by (*auto simp*: *hyperplane-cellcomplex-def*)

lemma *hyperplane-cellcomplex-Union*:
assumes $\bigwedge S.\ S \in \mathcal{C} \Longrightarrow$ *hyperplane-cellcomplex A S*
shows *hyperplane-cellcomplex A* $(\bigcup\ \mathcal{C})$
proof −
obtain $\mathcal{F}$ **where** $\mathcal{F}$: $\bigwedge S.\ S \in \mathcal{C} \Longrightarrow (\forall C \in \mathcal{F}\ S.$ *hyperplane-cell A C*$) \wedge S = \bigcup (\mathcal{F}\ S)$
by (*metis assms hyperplane-cellcomplex-def*)

show *?thesis*
unfolding *hyperplane-cellcomplex-def*
using $\mathcal{F}$ **by** (*fastforce intro*: *exI* [**where** $x{=}\bigcup\ (\mathcal{F}\ `\ \mathcal{C})$])
qed

lemma *hyperplane-cellcomplex-Un*:
$[\![$*hyperplane-cellcomplex A S*; *hyperplane-cellcomplex A T*$]\!]$
$\Longrightarrow$ *hyperplane-cellcomplex A* $(S \cup T)$
by (*smt* (*verit*) *Un-iff Union-Un-distrib hyperplane-cellcomplex-def*)

lemma *hyperplane-cellcomplex-UNIV* [*simp*]: *hyperplane-cellcomplex A UNIV*
by (*metis Union-hyperplane-cells hyperplane-cellcomplex-def mem-Collect-eq*)

lemma *hyperplane-cellcomplex-Inter*:
assumes $\bigwedge S.\ S \in \mathcal{C} \Longrightarrow$ *hyperplane-cellcomplex A S*
shows *hyperplane-cellcomplex A* $(\bigcap \mathcal{C})$
proof (*cases* $\mathcal{C} = \{\}$)
case *True*
then show *?thesis*
by *simp*
next
case *False*
obtain $\mathcal{F}$ **where** $\mathcal{F}$: $\bigwedge S.\ S \in \mathcal{C} \Longrightarrow (\forall\, C \in \mathcal{F}\ S.$ *hyperplane-cell A C*$) \wedge S = \bigcup(\mathcal{F}\ S)$
by (*metis assms hyperplane-cellcomplex-def*)
have $*$: $\mathcal{C} = (\lambda S.\ \bigcup(\mathcal{F}\ S))\ `\ \mathcal{C}$
using $\mathcal{F}$ **by** *force*
define U **where** $U \equiv \bigcup\ \{T \in \{\bigcap(g\ `\ \mathcal{C})\ |g.\ \forall\, S{\in}\mathcal{C}.\ g\ S \in \mathcal{F}\ S\}.\ T \neq \{\}\}$
have $\bigcap \mathcal{C} = \bigcup\{\bigcap(g\ `\ \mathcal{C})\ |g.\ \forall\, S{\in}\mathcal{C}.\ g\ S \in \mathcal{F}\ S\}$
using *False* $\mathcal{F}$ **unfolding** *Inter-over-Union* [*symmetric*]
by *blast*
also have $\ldots = U$
unfolding *U-def*
by *blast*
finally have $\bigcap \mathcal{C} = U$ **.**
have *hyperplane-cellcomplex A U*
using *False* $\mathcal{F}$ **unfolding** *U-def*
apply (*intro hyperplane-cellcomplex-Union hyperplane-cell-cellcomplex*)
by (*auto intro*!: *hyperplane-cell-Inter*)
then show *?thesis*
by (*simp add*: ‹$\bigcap \mathcal{C} = U$›)
qed

lemma *hyperplane-cellcomplex-Int*:
$[\![$*hyperplane-cellcomplex A S*; *hyperplane-cellcomplex A T*$]\!]$
$\Longrightarrow$ *hyperplane-cellcomplex A* $(S \cap T)$
using *hyperplane-cellcomplex-Inter* [*of* $\{S,T\}$] **by** *force*

lemma *hyperplane-cellcomplex-Compl*:

assumes *hyperplane-cellcomplex A S*
shows *hyperplane-cellcomplex A* $(- S)$
proof −
obtain $\mathcal{C}$ **where** $\mathcal{C}$: $\bigwedge C.\ C \in \mathcal{C} \Longrightarrow$ *hyperplane-cell A C* **and** $S = \bigcup\mathcal{C}$
by (*meson assms hyperplane-cellcomplex-def*)
have *hyperplane-cellcomplex A* $(\bigcap T \in \mathcal{C}.\ -T)$
proof (*intro hyperplane-cellcomplex-Inter*)
fix *C0*
assume $C0 \in$ *uminus* ' $\mathcal{C}$
then obtain C **where** C: $C0 = -C$ $C \in \mathcal{C}$
by *auto*
have ∗: $-C = \bigcup \{D.$ *hyperplane-cell A D* $\wedge D \neq C\}$ (**is** - = *?rhs*)
proof
show $- C \subseteq$ *?rhs*
using *hyperplane-cell* **by** *blast*
show *?rhs* $\subseteq - C$
by *clarify* (*meson* ‹$C \in \mathcal{C}$› $\mathcal{C}$ *disjnt-iff disjoint-hyperplane-cells*)
qed
then show *hyperplane-cellcomplex A C0*
by (*metis* (*no-types*, *lifting*) *C*(*1*) *hyperplane-cell-cellcomplex hyperplane-cellcomplex-Union mem-Collect-eq*)
qed
then show *?thesis*
by (*simp add*: ‹$S = \bigcup \mathcal{C}$› *uminus-Sup*)
qed

lemma *hyperplane-cellcomplex-diff*:
⟦*hyperplane-cellcomplex A S*; *hyperplane-cellcomplex A T*⟧
$\Longrightarrow$ *hyperplane-cellcomplex A* $(S - T)$
using *hyperplane-cellcomplex-Inter* [*of* $\{S,-T\}$]
by (*force simp*: *Diff-eq hyperplane-cellcomplex-Compl*)

lemma *hyperplane-cellcomplex-mono*:
assumes *hyperplane-cellcomplex A S* $A \subseteq B$
shows *hyperplane-cellcomplex B S*
proof −
obtain $\mathcal{C}$ **where** $\mathcal{C}$: $\bigwedge C.\ C \in \mathcal{C} \Longrightarrow$ *hyperplane-cell A C* **and** *eq*: $S = \bigcup\mathcal{C}$
by (*meson assms hyperplane-cellcomplex-def*)
show *?thesis*
unfolding *eq*
proof (*intro hyperplane-cellcomplex-Union*)
fix *C*
assume $C \in \mathcal{C}$
have $\bigwedge x.\ x \in C \Longrightarrow \exists D'.\ (\exists D.\ D' = D \cap C \wedge$ *hyperplane-cell* $(B - A)$ $D \wedge D \cap C \neq \{\}) \wedge x \in D'$
unfolding *hyperplane-cell-def* **by** *blast*
then
have *hyperplane-cellcomplex* $(A \cup (B - A))$ C
unfolding *hyperplane-cellcomplex-def hyperplane-cell-Un*

using $\mathcal{C}$ ‹$C \in \mathcal{C}$› **by** (*fastforce intro*!: *exI* [**where** x= $\{D \cap C$ |*D*. *hyperplane-cell* $(B - A)$ $D \wedge D \cap C \neq \{\}\}$])
moreover have $B = A \cup (B - A)$
using ‹$A \subseteq B$› **by** *auto*
ultimately show *hyperplane-cellcomplex B C* **by** *simp*
qed
qed

lemma *finite-hyperplane-cellcomplexes*:
assumes *finite A*
shows *finite* $\{C$. *hyperplane-cellcomplex A C*$\}$
proof −
have $\{C$. *hyperplane-cellcomplex A C*$\} \subseteq$ *image* $\bigcup$ $\{T$. $T \subseteq \{C$. *hyperplane-cell A C*$\}\}$
by (*force simp*: *hyperplane-cellcomplex-def subset-eq*)
with *finite-hyperplane-cells* **show** *?thesis*
by (*metis assms finite-Collect-subsets finite-surj*)
qed

lemma *finite-restrict-hyperplane-cellcomplexes*:
finite A $\Longrightarrow$ *finite* $\{C$. *hyperplane-cellcomplex A C* $\wedge$ *P C*$\}$
by (*simp add*: *finite-hyperplane-cellcomplexes*)

lemma *finite-set-of-hyperplane-cellcomplex*:
assumes *finite A* $\bigwedge C$. $C \in \mathcal{C} \Longrightarrow$ *hyperplane-cellcomplex A C*
shows *finite* $\mathcal{C}$
by (*metis assms finite-hyperplane-cellcomplexes mem-Collect-eq rev-finite-subset subsetI*)

lemma *cell-subset-cellcomplex*:
$[\![$*hyperplane-cell A C*; *hyperplane-cellcomplex A S*$]\!] \Longrightarrow C \subseteq S \longleftrightarrow$ ~ *disjnt C S*
by (*smt* (*verit*) *Union-iff disjnt-iff disjnt-subset1 disjoint-hyperplane-cells-eq hyperplane-cellcomplex-def subsetI*)

7.3 Euler characteristic

definition *Euler-characteristic* :: (*'a*::*euclidean-space* × *real*) *set* ⇒ *'a set* ⇒ *int*
where *Euler-characteristic A S* ≡
$(\sum C \mid$ *hyperplane-cell A C* $\wedge C \subseteq S$. (-1) ^ *nat* (*aff-dim C*))

lemma *Euler-characteristic-empty* [*simp*]: *Euler-characteristic A* $\{\} = 0$
by (*simp add*: *sum.neutral Euler-characteristic-def*)

lemma *Euler-characteristic-cell-Union*:
assumes $\bigwedge C$. $C \in \mathcal{C} \Longrightarrow$ *hyperplane-cell A C*
shows *Euler-characteristic A* $(\bigcup \mathcal{C}) = (\sum C \in \mathcal{C}$. $(- 1)$ ^ *nat* (*aff-dim C*))
proof −
have $\bigwedge x$. $[\![$*hyperplane-cell A x*; $x \subseteq \bigcup \mathcal{C}]\!] \Longrightarrow x \in \mathcal{C}$
by (*metis assms disjnt-Union1 disjnt-subset1 disjoint-hyperplane-cells-eq*)

then have $\{C.\ \textit{hyperplane-cell}\ A\ C \land C \subseteq \bigcup \mathcal{C}\} = \mathcal{C}$
  by (*auto simp*: *assms*)
then show *?thesis*
  by (*auto simp*: *Euler-characteristic-def*)
qed

lemma *Euler-characteristic-cell*:
  $\textit{hyperplane-cell}\ A\ C \Longrightarrow \textit{Euler-characteristic}\ A\ C = (-1)\ \widehat{}\ (\textit{nat}(\textit{aff-dim}\ C))$
  using *Euler-characteristic-cell-Union* [*of* $\{C\}$] **by** *force*

lemma *Euler-characteristic-cellcomplex-Un*:
  assumes *finite A hyperplane-cellcomplex A S*
    and *AT*: *hyperplane-cellcomplex A T* **and** *disjnt S T*
  shows $\textit{Euler-characteristic}\ A\ (S \cup T) =$
      $\textit{Euler-characteristic}\ A\ S + \textit{Euler-characteristic}\ A\ T$
proof –
  have $*$: $\{C.\ \textit{hyperplane-cell}\ A\ C \land C \subseteq S \cup T\} =$
      $\{C.\ \textit{hyperplane-cell}\ A\ C \land C \subseteq S\} \cup \{C.\ \textit{hyperplane-cell}\ A\ C \land C \subseteq T\}$
    using *cell-subset-cellcomplex* [*OF - AT*] **by** (*auto simp*: *disjnt-iff*)
  have $**$: $\{C.\ \textit{hyperplane-cell}\ A\ C \land C \subseteq S\} \cap \{C.\ \textit{hyperplane-cell}\ A\ C \land C \subseteq T\} = \{\}$
    using *assms cell-subset-cellcomplex disjnt-subset1* **by** *fastforce*
  show *?thesis*
  unfolding *Euler-characteristic-def*
  by (*simp add*: *finite-restrict-hyperplane-cells assms* $*$ $**$ *flip*: *sum.union-disjoint*)
qed

lemma *Euler-characteristic-cellcomplex-Union*:
  assumes *finite A*
    and $\mathcal{C}$: $\bigwedge C.\ C \in \mathcal{C} \Longrightarrow \textit{hyperplane-cellcomplex}\ A\ C$ *pairwise disjnt* $\mathcal{C}$
  shows $\textit{Euler-characteristic}\ A\ (\bigcup \mathcal{C}) = \textit{sum}\ (\textit{Euler-characteristic}\ A)\ \mathcal{C}$
proof –
  have *finite* $\mathcal{C}$
    using *assms finite-set-of-hyperplane-cellcomplex* **by** *blast*
  then show *?thesis*
    using $\mathcal{C}$
  proof (*induction rule*: *finite-induct*)
    case *empty*
    then show *?case*
      by *auto*
  next
    case (*insert C* $\mathcal{C}$)
    then obtain *disjoint* $\mathcal{C}$ *disjnt C* $(\bigcup \mathcal{C})$
      by (*metis disjnt-Union2 pairwise-insert*)
    with *insert* **show** *?case*
    by (*simp add*: *Euler-characteristic-cellcomplex-Un hyperplane-cellcomplex-Union* ‹*finite A*›)
  qed
qed

lemma *Euler-characteristic*:
  fixes A :: $('n$::*euclidean-space* $*$ *real*) *set*
  assumes *finite A*
  shows *Euler-characteristic A S* =
    $(\sum d = 0..DIM('n).\ (-1)\ \hat{}\ d * int\ (card\ \{C.\ hyperplane\text{-}cell\ A\ C \wedge C \subseteq S \wedge aff\text{-}dim\ C = int\ d\}))$
    (**is** - = *?rhs*)
proof −
  have $\bigwedge T.\ [\![hyperplane\text{-}cell\ A\ T;\ T \subseteq S]\!] \Longrightarrow aff\text{-}dim\ T \in \{0..DIM('n)\}$
    by (*metis atLeastAtMost-iff nle-le order.strict-iff-not aff-dim-negative-iff nonempty-hyperplane-cell aff-dim-le-DIM*)
  then have $*$: $aff\text{-}dim\ `\ \{C.\ hyperplane\text{-}cell\ A\ C \wedge C \subseteq S\} \subseteq int\ `\ \{0..DIM('n)\}$
    by (*auto simp*: *image-int-atLeastAtMost*)
  have $Euler\text{-}characteristic\ A\ \ S = (\sum y{\in}int\ `\ \{0..DIM('n)\}.$
    $\sum C{\in}\{x.\ hyperplane\text{-}cell\ A\ x \wedge x \subseteq S \wedge aff\text{-}dim\ x = y\}.\ (-\ 1)\ \hat{}\ nat\ y)$
    using *sum.group* [*of* $\{C.\ hyperplane\text{-}cell\ A\ C \wedge C \subseteq S\}$ $int\ `\ \{0..DIM('n)\}$ *aff-dim* $\lambda C.\ (-1{::}int)\ \hat{}\ nat(aff\text{-}dim\ C)$, *symmetric*]
    by (*simp add*: *assms Euler-characteristic-def finite-restrict-hyperplane-cells* $*$)
  also have ... = *?rhs*
    by (*simp add*: *sum.reindex mult-of-nat-commute*)
  finally show *?thesis* **.**
qed

7.4 Show that the characteristic is invariant w.r.t. hyperplane arrangement.

lemma *hyperplane-cells-distinct-lemma*:
  $\{x.\ a \cdot x = b\} \cap \{x.\ a \cdot x < b\} = \{\} \wedge$
    $\{x.\ a \cdot x = b\} \cap \{x.\ a \cdot x > b\} = \{\} \wedge$
    $\{x.\ a \cdot x < b\} \cap \{x.\ a \cdot x = b\} = \{\} \wedge$
    $\{x.\ a \cdot x < b\} \cap \{x.\ a \cdot x > b\} = \{\} \wedge$
    $\{x.\ a \cdot x > b\} \cap \{x.\ a \cdot x = b\} = \{\} \wedge$
    $\{x.\ a \cdot x > b\} \cap \{x.\ a \cdot x < b\} = \{\}$
  by *auto*

proposition *Euler-characterstic-lemma*:
  assumes *finite A* **and** *hyperplane-cellcomplex A S*
  shows *Euler-characteristic* (*insert h A*) *S* = *Euler-characteristic A S*
proof −
  obtain $\mathcal{C}$ **where** $\mathcal{C}$: $\bigwedge C.\ C \in \mathcal{C} \Longrightarrow hyperplane\text{-}cell\ A\ C$ **and** $S = \bigcup \mathcal{C}$
      and *pairwise disjnt* $\mathcal{C}$
    by (*meson assms hyperplane-cellcomplex-def pairwise-disjoint-hyperplane-cells*)
  obtain *a b* **where** $h = (a,b)$
    by *fastforce*
  have $\bigwedge C.\ C \in \mathcal{C} \Longrightarrow hyperplane\text{-}cellcomplex\ A\ C \wedge hyperplane\text{-}cellcomplex\ (insert\ (a,b)\ A)\ C$
    by (*meson* $\mathcal{C}$ *hyperplane-cell-cellcomplex hyperplane-cellcomplex-mono subset-insertI*)

```
  moreover
  have sum (Euler-characteristic (insert (a,b) A)) 𝒞 = sum (Euler-characteristic
A) 𝒞
  proof (rule sum.cong [OF refl])
    fix C
    assume C ∈ 𝒞
    have Euler-characteristic (insert (a, b) A) C = (−1) ^ nat(aff-dim C)
    proof (cases hyperplane-cell (insert (a,b) A) C)
      case True
      then show ?thesis
        using Euler-characteristic-cell by blast
    next
      case False
      with 𝒞[OF ‹C ∈ 𝒞›] have a ≠ 0
          by (smt (verit, ccfv-threshold) hyperplane-cell-Un hyperplane-cell-empty
hyperplane-cell-singleton insert-is-Un sup-bot-left)
      have convex C
        using ‹hyperplane-cell A C› hyperplane-cell-convex by blast
      define r where r ≡ (∑ D∈{C' ∩ C |C'. hyperplane-cell {(a, b)} C' ∧ C' ∩
C ≠ {}}. (−1::int) ^ nat (aff-dim D))
      have Euler-characteristic (insert (a, b) A) C
          = (∑ D | (D ≠ {} ∧
                  (∃ C1 C2. hyperplane-cell {(a, b)} C1 ∧ hyperplane-cell A C2 ∧
D = C1 ∩ C2)) ∧ D ⊆ C.
             (− 1) ^ nat (aff-dim D))
      unfolding r-def Euler-characteristic-def insert-is-Un [of - A] hyperplane-cell-Un
..
      also have ... = r
        unfolding r-def
        apply (rule sum.cong [OF - refl])
        using ‹hyperplane-cell A C› disjoint-hyperplane-cells disjnt-iff
        by (smt (verit, ccfv-SIG) Collect-cong Int-iff disjoint-iff subsetD subsetI)
      also have ... = (−1) ^ nat(aff-dim C)
      proof −
        have C ≠ {}
          using ‹hyperplane-cell A C› by auto
        show ?thesis
        proof (cases C ⊆ {x. a · x < b} ∨ C ⊆ {x. a · x > b} ∨ C ⊆ {x. a · x =
b})
          case Csub: True
          with ‹C ≠ {}› have r = sum (λc. (−1) ^ nat (aff-dim c)) {C}
            unfolding r-def
            apply (intro sum.cong [OF - refl])
            by (auto simp: ‹a ≠ 0› hyperplane-cell-singleton)
          also have ... = (−1) ^ nat(aff-dim C)
            by simp
          finally show ?thesis .
        next
          case False
```

```
then obtain u v where uv: u ∈ C ¬ a · u < b v ∈ C ¬ a · v > b
  by blast
have CInt-ne: C ∩ {x. a · x = b} ≠ {}
proof (cases a · u = b ∨ a · v = b)
  case True
  with uv show ?thesis
    by blast
next
  case False
  have a · v < a · u
    using False uv by auto
  define w where w ≡ v + ((b − a · v) / (a · u − a · v)) *R (u − v)
  have **: v + a *R (u − v) = (1 − a) *R v + a *R u for a
    by (simp add: algebra-simps)
  have w ∈ C
    unfolding w-def **
  proof (intro convexD-alt)
  qed (use ‹a · v < a · u› ‹convex C› uv in auto)
  moreover have w ∈ {x. a · x = b}
 using ‹a · v < a · u› by (simp add: w-def inner-add-right inner-diff-right)
  ultimately show ?thesis
    by blast
qed
have Cab: C ∩ {x. a · x < b} ≠ {} ∧ C ∩ {x. b < a · x} ≠ {}
proof −
  obtain u v where u ∈ C a · u = b v ∈ C a · v ≠ b u≠v
    using False ‹C ∩ {x. a · x = b} ≠ {}› by blast
  have openin (subtopology euclidean (affine hull C)) C
     using ‹hyperplane-cell A C› ‹finite A› hyperplane-cell-relatively-open
by blast
  then obtain ε where 0 < ε
       and ε: ⋀x'. ⟦x' ∈ affine hull C; dist x' u < ε⟧ ⟹ x' ∈ C
    by (meson ‹u ∈ C› openin-euclidean-subtopology-iff)
  define ξ where ξ ≡ u − (ε / 2 / norm (v − u)) *R (v − u)
  have ξ ∈ C
  proof (rule ε)
    show ξ ∈ affine hull C
      by (simp add: ξ-def ‹u ∈ C› ‹v ∈ C› hull-inc mem-affine-3-minus2)
  qed (use ξ-def ‹0 < ε› in force)
  consider a · v < b | a · v > b
    using ‹a · v ≠ b› by linarith
  then show ?thesis
  proof cases
    case 1
    moreover have ξ ∈ {x. b < a · x}
      using 1 ‹0 < ε› ‹a · u = b› divide-less-cancel
      by (fastforce simp: ξ-def algebra-simps)
    ultimately show ?thesis
      using ‹v ∈ C› ‹ξ ∈ C› by blast
```

next
case *2*
moreover have $\xi \in \{x.\ b > a \cdot x\}$
using *2* ‹$0 < \varepsilon$› ‹$a \cdot u = b$› *divide-less-cancel*
by (*fastforce simp*: *ξ-def algebra-simps*)
ultimately show *?thesis*
using ‹$v \in C$› ‹$\xi \in C$› **by** *blast*
qed
qed
have $r = (\sum C{\in}\{\{x.\ a \cdot x = b\} \cap C,\ \{x.\ b < a \cdot x\} \cap C,\ \{x.\ a \cdot x < b\} \cap C\}.$
$(-\ 1)\ \widehat{}\ nat\ (aff\text{-}dim\ C))$
unfolding *r-def*
proof (*intro sum.cong* [*OF - refl*] *equalityI*)
show $\{\{x.\ a \cdot x = b\} \cap C,\ \{x.\ b < a \cdot x\} \cap C,\ \{x.\ a \cdot x < b\} \cap C\}$
$\subseteq \{C' \cap C\ |C'.\ hyperplane\text{-}cell\ \{(a,\ b)\}\ C' \wedge C' \cap C \neq \{\}\}$
apply *clarsimp*
using *Cab Int-commute* ‹$C \cap \{x.\ a \cdot x = b\} \neq \{\}$› *hyperplane-cell-singleton* ‹$a \neq 0$›
by *metis*
qed (*auto simp*: ‹$a \neq 0$› *hyperplane-cell-singleton*)
also have $\ldots = (-1)\ \widehat{}\ nat\ (aff\text{-}dim\ (C \cap \{x.\ a \cdot x = b\}))$
$+ (-1)\ \widehat{}\ nat\ (aff\text{-}dim\ (C \cap \{x.\ b < a \cdot x\}))$
$+ (-1)\ \widehat{}\ nat\ (aff\text{-}dim\ (C \cap \{x.\ a \cdot x < b\}))$
using *hyperplane-cells-distinct-lemma* [*of a b*] *Cab*
by (*auto simp*: *sum.insert-if Int-commute Int-left-commute*)
also have $\ldots = (-\ 1)\ \widehat{}\ nat\ (aff\text{-}dim\ C)$
proof −
have $*$: $aff\text{-}dim\ (C \cap \{x.\ a \cdot x < b\}) = aff\text{-}dim\ C \wedge aff\text{-}dim\ (C \cap \{x.\ a \cdot x > b\}) = aff\text{-}dim\ C$
by (*metis Cab open-halfspace-lt open-halfspace-gt aff-dim-affine-hull affine-hull-convex-Int-open*[*OF* ‹*convex C*›])
obtain *S T* **where** *open S affine T* **and** *Ceq*: $C = S \cap T$
by (*meson* ‹*hyperplane-cell A C*› ‹*finite A*› *hyperplane-cell-Int-open-affine*)
have *affine hull C = affine hull T*
by (*metis Ceq* ‹$C \neq \{\}$› ‹*affine T*› ‹*open S*› *affine-hull-affine-Int-open inf-commute*)
moreover
have $T \cap (\{x.\ a \cdot x = b\} \cap S) \neq \{\}$
using *Ceq* ‹$C \cap \{x.\ a \cdot x = b\} \neq \{\}$› **by** *blast*
then have $affine\ hull\ (C \cap \{x.\ a \cdot x = b\}) = affine\ hull\ (T \cap \{x.\ a \cdot x = b\})$
using *affine-hull-affine-Int-open*[*of* $T \cap \{x.\ a \cdot x = b\}$ *S*]
by (*simp add*: *Ceq Int-ac* ‹*affine T*› ‹*open S*› *affine-Int affine-hyperplane*)
ultimately have $aff\text{-}dim\ (affine\ hull\ C) = aff\text{-}dim(affine\ hull\ (C \cap \{x.\ a \cdot x = b\})) + 1$
using *CInt-ne False Ceq*
by (*auto simp*: *aff-dim-affine-Int-hyperplane* ‹*affine T*›)
moreover have $0 \leq aff\text{-}dim\ (C \cap \{x.\ a \cdot x = b\})$

by (*metis CInt-ne aff-dim-negative-iff linorder-not-le*)
ultimately show *?thesis*
by (*simp add*: ∗ *nat-add-distrib*)
qed
finally show *?thesis* **.**
qed
qed
finally show *Euler-characteristic* (*insert* (*a*, *b*) *A*) *C* = (−*1*) ^ *nat*(*aff-dim* *C*) **.**
qed
then show *Euler-characteristic* (*insert* (*a*, *b*) *A*) *C* = (*Euler-characteristic A* *C*)
by (*simp add*: *Euler-characteristic-cell* $\mathcal{C}$ ‹*C* ∈ $\mathcal{C}$›)
qed
ultimately show *?thesis*
by (*simp add*: *Euler-characteristic-cellcomplex-Union* ‹*S* = ⋃ $\mathcal{C}$› ‹*disjoint* $\mathcal{C}$› ‹*h* = (*a*, *b*)› *assms*(*1*))
qed

lemma *Euler-characterstic-invariant-aux*:
assumes *finite B finite A hyperplane-cellcomplex A S*
shows *Euler-characteristic* (*A* ∪ *B*) *S* = *Euler-characteristic A S*
using *assms*
by (*induction rule*: *finite-induct*) (*auto simp*: *Euler-characterstic-lemma hyperplane-cellcomplex-mono*)

lemma *Euler-characterstic-invariant*:
assumes *finite A finite B hyperplane-cellcomplex A S hyperplane-cellcomplex B S*
shows *Euler-characteristic A S* = *Euler-characteristic B S*
by (*metis Euler-characterstic-invariant-aux assms sup-commute*)

lemma *Euler-characteristic-inclusion-exclusion*:
assumes *finite A finite* $\mathcal{S}$ ⋀*K*. *K* ∈ $\mathcal{S}$ ⟹ *hyperplane-cellcomplex A K*
shows *Euler-characteristic A* (⋃ $\mathcal{S}$) = (∑$\mathcal{T}$ | $\mathcal{T}$ ⊆ $\mathcal{S}$ ∧ $\mathcal{T}$ ≠ {}. (− *1*) ^(*card* $\mathcal{T}$ + *1*) ∗ *Euler-characteristic A* (⋂$\mathcal{T}$))
proof −
interpret *Incl-Excl hyperplane-cellcomplex A Euler-characteristic A*
proof
show *Euler-characteristic A* (*S* ∪ *T*) = *Euler-characteristic A S* + *Euler-characteristic A T*
if *hyperplane-cellcomplex A S* **and** *hyperplane-cellcomplex A T* **and** *disjnt S T* **for** *S T*
using *that Euler-characteristic-cellcomplex-Un assms*(*1*) **by** *blast*
qed (*use hyperplane-cellcomplex-Int hyperplane-cellcomplex-Un hyperplane-cellcomplex-diff* **in** *auto*)
show *?thesis*
using *restricted assms* **by** *blast*

qed

7.5 Euler-type relation for full-dimensional proper polyhedral cones

```
lemma Euler-polyhedral-cone:
  fixes S :: 'n::euclidean-space set
  assumes polyhedron S conic S and intS: interior S ≠ {} and S ≠ UNIV
  shows (∑ d = 0..DIM('n). (− 1) ^ d ∗ int (card {f. f face-of S ∧ aff-dim f =
int d})) = 0  (is ?lhs = 0)
proof −
  have [simp]: affine hull S = UNIV
    by (simp add: affine-hull-nonempty-interior intS)
  with ‹polyhedron S›
  obtain H where finite H
    and Seq: S = ⋂H
    and Hex: ⋀h. h∈H ⟹ ∃ a b. a≠0 ∧ h = {x. a · x ≤ b}
    and Hsub: ⋀𝒢. 𝒢 ⊂ H ⟹ S ⊂ ⋂𝒢
    by (fastforce simp: polyhedron-Int-affine-minimal)
  have 0 ∈ S
    using assms(2) conic-contains-0 intS interior-empty by blast
  have ∗: ∃ a. a≠0 ∧ h = {x. a · x ≤ 0} if h ∈ H for h
  proof −
    obtain a b where a≠0 and ab: h = {x. a · x ≤ b}
      using Hex [OF ‹h ∈ H›] by blast
    have 0 ∈ ⋂H
      using Seq ‹0 ∈ S› by force
    then have 0 ∈ h
      using that by blast
    consider b=0 | b < 0 | b > 0
      by linarith
    then
    show ?thesis
    proof cases
      case 1
      then show ?thesis
        using ‹a ≠ 0› ab by blast
    next
      case 2
      then show ?thesis
        using ‹0 ∈ h› ab by auto
    next
      case 3
      have S ⊂ ⋂(H − {h})
        using Hsub [of H − {h}] that by auto
      then obtain x where x: x ∈ ⋂(H − {h}) and x ∉ S
        by auto
      define ε where ε ≡ min (1/2) (b / (a · x))
      have b < a · x
```

```
        using ‹x ∉ S› ab x by (fastforce simp: ‹S = ⋂ H›)
      with 3 have 0 < a · x
        by auto
      with 3 have 0 < ε
        by (simp add: ε-def)
      have ε < 1
        using ε-def by linarith
      have ε * (a · x) ≤ b
        unfolding ε-def using ‹0 < a · x› pos-le-divide-eq by fastforce
      have x = inverse ε *R ε *R x
        using ‹0 < ε› by force
      moreover
      have ε *R x ∈ S
      proof −
        have ε *R x ∈ h
          by (simp add: ‹ε * (a · x) ≤ b› ab)
        moreover have ε *R x ∈ ⋂(H − {h})
        proof −
          have ε *R x ∈ k if x ∈ k k ∈ H k ≠ h for k
          proof −
            obtain a' b' where a'≠0 k = {x. a' · x ≤ b'}
              using Hex ‹k ∈ H› by blast
            have (0 ≤ a' · x ⟹ a' · ε *R x ≤ a' · x)
              by (metis ‹ε < 1› inner-scaleR-right order-less-le pth-1 real-scaleR-def
scaleR-right-mono)
            moreover have (0 ≤ −(a' · x) ⟹ 0 ≤ −(a' · ε *R x))
              using ‹0 < ε› mult-le-0-iff order-less-imp-le by auto
            ultimately
            have a' · x ≤ b' ⟹ a' · ε *R x ≤ b'
             by (smt (verit) InterD ‹0 ∈ ⋂ H› ‹k = {x. a' · x ≤ b'}› inner-zero-right
mem-Collect-eq that(2))
            then show ?thesis
              using ‹k = {x. a' · x ≤ b'}› ‹x ∈ k› by fastforce
          qed
          with x show ?thesis
            by blast
        qed
        ultimately show ?thesis
          using Seq by blast
      qed
      with ‹conic S› have inverse ε *R ε *R x ∈ S
      by (meson ‹0 < ε› conic-def inverse-nonnegative-iff-nonnegative order-less-le)
      ultimately show ?thesis
        using ‹x ∉ S› by presburger
    qed
  qed
  then obtain fa where fa: ⋀h. h ∈ H ⟹ fa h ≠ 0 ∧ h = {x. fa h · x ≤ 0}
    by metis
  define fa-le-0 where fa-le-0 ≡ λh. {x. fa h · x ≤ 0}
```

have *fa'*: $\bigwedge h.\ h \in H \Longrightarrow$ *fa-le-0* $h = h$
using *fa fa-le-0-def* **by** *blast*
define *A* **where** $A \equiv (\lambda h.\ (fa\ h, 0{::}real))$ ‘ H
have *finite A*
using ‹*finite H*› **by** (*simp add*: *A-def*)
then have *?lhs* = *Euler-characteristic A S*
proof −
have [*simp*]: *card* {*f*. *f face-of S* ∧ *aff-dim f* = *int d*} = *card* {*C*. *hyperplane-cell A C* ∧ *C* ⊆ *S* ∧ *aff-dim C* = *int d*}
if *finite A* **and** $d \le$ *card* (*Basis*::'*n set*)
for *d* :: *nat*
proof (*rule bij-betw-same-card*)
have *hyper1*: *hyperplane-cell A* (*rel-interior f*) ∧ *rel-interior f* ⊆ *S* ∧ *aff-dim* (*rel-interior f*) = *d* ∧ *closure* (*rel-interior f*) = *f*
if *f face-of S aff-dim f* = *d* **for** *f*
proof −
have *1*: *closure*(*rel-interior f*) = *f*
proof −
have *closure*(*rel-interior f*) = *closure f*
by (*meson convex-closure-rel-interior face-of-imp-convex that*(*1*))
also have ... = *f*
by (*meson assms*(*1*) *closure-closed face-of-polyhedron-polyhedron polyhedron-imp-closed that*(*1*))
finally show *?thesis* **.**
qed
then have *2*: *aff-dim* (*rel-interior f*) = *d*
by (*metis closure-aff-dim that*(*2*))
have $f \ne \{\}$
using *aff-dim-negative-iff* [*of f*] **by** (*simp add*: *that*(*2*))
obtain *J0* **where** *J0* ⊆ *H* **and** *J0*: $f = \bigcap$ (*fa-le-0* ‘ *H*) ∩ ($\bigcap h \in J0$. {*x*. *fa h* · *x* = *0*})
proof (*cases f* = *S*)
case *True*
have $S = \bigcap$ (*fa-le-0* ‘ *H*)
using *Seq fa* **by** (*auto simp*: *fa-le-0-def*)
then show *?thesis*
using *True that* **by** *blast*
next
case *False*
have *fexp*: $f = \bigcap$ {*S* ∩ {*x*. *fa h* · *x* = *0*} | *h*. *h* ∈ *H* ∧ *f* ⊆ *S* ∩ {*x*. *fa h* · *x* = *0*}}
proof (*rule face-of-polyhedron-explicit*)
show *S* = *affine hull S* ∩ $\bigcap$ *H*
by (*simp add*: *Seq hull-subset inf.absorb2*)
qed (*auto simp*: *False* ‹$f \ne \{\}$› ‹*f face-of S*› ‹*finite H*› *Hsub fa*)
show *?thesis*
proof
have ∗: $\bigwedge x\ h.$ ⟦$x \in f$; $h \in H$⟧ $\Longrightarrow$ *fa h* · $x \le 0$
using *Seq fa face-of-imp-subset* ‹*f face-of S*› **by** *fastforce*

show $f = \bigcap$ $(\textit{fa-le-0}$ ` $H) \cap (\bigcap h \in \{h \in H.\ f \subseteq S \cap \{x.\ \textit{fa}\ h \cdot x = \textit{0}\}\}.\ \{x.\ \textit{fa}\ h \cdot x = \textit{0}\})$
(**is** $f = \textit{?I}$)
proof
show $f \subseteq \textit{?I}$
using ‹*f face-of S*› *fa face-of-imp-subset* **by** (*force simp*: ∗ *fa-le-0-def*)
show $\textit{?I} \subseteq f$
apply (*subst* (*2*) *fexp*)
apply (*clarsimp simp*: ∗ *fa-le-0-def*)
by (*metis Inter-iff Seq fa mem-Collect-eq*)
qed
qed *blast*
qed
define H' **where** $H' = (\lambda h.\ \{x.\ -(\textit{fa}\ h) \cdot x \le \textit{0}\})$ ` H
have $\exists J.\ \textit{finite}\ J \wedge J \subseteq H \cup H' \wedge f = \textit{affine hull}\ f \cap \bigcap J$
proof (*intro exI conjI*)
let $\textit{?J} = H \cup \textit{image}\ (\lambda h.\ \{x.\ -(\textit{fa}\ h) \cdot x \le \textit{0}\})\ \textit{J0}$
show *finite* (*?J*::′*n set set*)
using ‹$\textit{J0} \subseteq H$› ‹*finite H*› *finite-subset* **by** *fastforce*
show $\textit{?J} \subseteq H \cup H'$
using ‹$\textit{J0} \subseteq H$› **by** (*auto simp*: H'-*def*)
have $f = \bigcap \textit{?J}$
proof
show $f \subseteq \bigcap \textit{?J}$
unfolding *J0* **by** (*auto simp*: *fa*′)
have $\bigwedge x\ j.\ [\![j \in \textit{J0};\ \forall h{\in}H.\ x \in h;\ \forall j{\in}\textit{J0}.\ \textit{0} \le \textit{fa}\ j \cdot x]\!] \Longrightarrow \textit{fa}\ j \cdot x = \textit{0}$
by (*metis* ‹$\textit{J0} \subseteq H$› *fa in-mono inf.absorb2 inf.orderE mem-Collect-eq*)
then show $\bigcap \textit{?J} \subseteq f$
unfolding *J0* **by** (*auto simp*: *fa*′)
qed
then show $f = \textit{affine hull}\ f \cap \bigcap \textit{?J}$
by (*simp add*: *Int-absorb1 hull-subset*)
qed
then have ∗∗: $\exists n\ J.\ \textit{finite}\ J \wedge \textit{card}\ J = n \wedge J \subseteq H \cup H' \wedge f = \textit{affine hull}\ f \cap \bigcap J$
by *blast*
obtain *J nJ* **where** *J*: *finite J card J* = *nJ* $J \subseteq H \cup H'$ **and** *feq*: $f = \textit{affine hull}\ f \cap \bigcap J$
and *minJ*: $\bigwedge m\ J'.\ [\![\textit{finite}\ J';\ m < nJ;\ \textit{card}\ J' = m;\ J' \subseteq H \cup H']\!] \Longrightarrow f \ne \textit{affine hull}\ f \cap \bigcap J'$
using *exists-least-iff* [*THEN iffD1*, *OF* ∗∗] **by** *metis*
have *FF*: $f \subset (\textit{affine hull}\ f \cap \bigcap J')$ **if** $J' \subset J$ **for** J'
proof −
have $f \ne \textit{affine hull}\ f \cap \bigcap J'$
using *minJ*
by (*metis J finite-subset psubset-card-mono psubset-imp-subset psubset-subset-trans that*)
then show *?thesis*
by (*metis Int-subset-iff Inter-Un-distrib feq hull-subset inf-sup-ord*(*2*)

psubsetI sup.absorb4 that)
qed
have $\exists a.\ \{x.\ a \cdot x \le 0\} = h \wedge (h \in H \wedge a = fa\ h \vee (\exists h'.\ h' \in H \wedge a = -(fa\ h')))$
if $h \in J$ **for** h
proof −
have $h \in H \cup H'$
using ‹$J \subseteq H \cup H'$› *that* **by** *blast*
then show *?thesis*
proof
show *?thesis* **if** $h \in H$
using *that fa* **by** *blast*
next
assume $h \in H'$
then obtain h' **where** $h' \in H$ $h = \{x.\ 0 \le fa\ h' \cdot x\}$
by (*auto simp*: *H′-def*)
then show *?thesis*
by (*force simp*: *intro*!: *exI*[**where** *x*=− (*fa h′*)])
qed
qed
then obtain *ga*
where *ga-h*: $\bigwedge h.\ h \in J \Longrightarrow h = \{x.\ ga\ h \cdot x \le 0\}$
and *ga-fa*: $\bigwedge h.\ h \in J \Longrightarrow h \in H \wedge ga\ h = fa\ h \vee (\exists h'.\ h' \in H \wedge ga\ h = -(fa\ h'))$
by *metis*
have *3*: *hyperplane-cell A* (*rel-interior f*)
proof −
have *D*: $rel\text{-}interior\ f = \{x \in f.\ \forall h{\in}J.\ ga\ h \cdot x < 0\}$
proof (*rule rel-interior-polyhedron-explicit* [*OF* ‹*finite J*› *feq*])
show $ga\ h \ne 0 \wedge h = \{x.\ ga\ h \cdot x \le 0\}$ **if** $h \in J$ **for** h
using *that fa ga-fa ga-h* **by** *force*
qed (*auto simp*: *FF*)
have *H*: $h \in H \wedge ga\ h = fa\ h$ **if** $h \in J$ **for** h
proof −
obtain *z* **where** *z*: $z \in rel\text{-}interior\ f$
using *1* ‹$f \ne \{\}$› **by** *force*
then have $z \in f \wedge z \in S$
using *D* ‹*f face-of S*› *face-of-imp-subset* **by** *blast*
then show *?thesis*
using *ga-fa* [*OF that*]
by (*smt* (*verit*, *del-insts*) *D InterE Seq fa inner-minus-left mem-Collect-eq that z*)
qed
then obtain *K* **where** $K \subseteq H$
and *K*: $f = \bigcap\ (fa\text{-}le\text{-}0\ `\ H) \cap (\bigcap h \in K.\ \{x.\ fa\ h \cdot x = 0\})$
using *J0* ‹$J0 \subseteq H$› **by** *blast*
have *E*: $rel\text{-}interior\ f = \{x.\ (\forall h \in H.\ fa\ h \cdot x \le 0) \wedge (\forall h \in K.\ fa\ h \cdot x = 0) \wedge (\forall h \in J.\ ga\ h \cdot x < 0)\}$
unfolding *D* **by** (*simp add*: *K fa-le-0-def*)

```
    have relif: rel-interior f ≠ {}
      using 1 ‹f ≠ {}› by force
    with E have disjnt J K
      using H disjnt-iff by fastforce
    define IFJK where IFJK ≡ λh. if h ∈ J then {x. fa h · x < 0}
                    else if h ∈ K then {x. fa h · x = 0}
                    else if rel-interior f ⊆ {x. fa h · x = 0}
                    then {x. fa h · x = 0}
                    else {x. fa h · x < 0}
    have relint-f: rel-interior f = ⋂(IFJK ‘ H)
    proof
      have A: False
       if x: x ∈ rel-interior f and y: y ∈ rel-interior f and less0: fa h · y < 0
         and fa0:  fa h · x = 0 and h ∈ H h ∉ J h ∉ K  for x h y
      proof −
        obtain ε where x ∈ f ε>0
          and ε: ⋀t. ⟦dist x t ≤ ε; t ∈ affine hull f⟧ ⟹ t ∈ f
          using x by (force simp: mem-rel-interior-cball)
        then have y ≠ x
          using fa0 less0 by force
        define x' where x' ≡ x + (ε / norm(y − x)) *R (x − y)
        have x ∈ affine hull f ∧ y ∈ affine hull f
          by (metis ‹x ∈ f› hull-inc mem-rel-interior-cball y)
        moreover have dist x x' ≤ ε
          using ‹0 < ε› ‹y ≠ x› by (simp add: x'-def divide-simps dist-norm
norm-minus-commute)
        ultimately have x' ∈ f
          by (simp add: ε mem-affine-3-minus x'-def)
        have x' ∈ S
          using ‹f face-of S› ‹x' ∈ f› face-of-imp-subset by auto
        then have x' ∈ h
          using Seq that(5) by blast
        then have x' ∈ {x. fa h · x ≤ 0}
          using fa that(5) by blast
        moreover have ε / norm (y − x) ∗ −(fa h · y) > 0
          using  ‹0 < ε› ‹y ≠ x› less0 by (simp add: field-split-simps)
        ultimately show ?thesis
          by (simp add: x'-def fa0 inner-diff-right inner-right-distrib)
      qed
      show rel-interior f ⊆ ⋂(IFJK ‘ H)
        unfolding IFJK-def by (smt (verit, ccfv-SIG) A E H INT-I in-mono
mem-Collect-eq subsetI)
      show ⋂(IFJK ‘ H) ⊆ rel-interior f
        using ‹K ⊆ H› ‹disjnt J K›
        apply (clarsimp simp add: ball-Un E H disjnt-iff IFJK-def)
        apply (smt (verit, del-insts) IntI Int-Collect subsetD)
        done
    qed
    obtain z where zrelf: z ∈ rel-interior f
```

using *relif* **by** *blast*
moreover
have H: $z \in$ *IFJK h* $\Longrightarrow$ ($x \in$ *IFJK h*) = (*hyperplane-side* (*fa h*, *0*) z = *hyperplane-side* (*fa h*, *0*) x) **for** *h x*
using *zrelf* **by** (*auto simp*: *IFJK-def hyperplane-side-def sgn-if split*: *if-split-asm*)
then have $z \in \bigcap$(*IFJK* ‘ H) $\Longrightarrow$ ($x \in \bigcap$(*IFJK* ‘ H)) = *hyperplane-equiv* *A z x* **for** *x*
unfolding *A-def Inter-iff hyperplane-equiv-def ball-simps* **using** *H* **by** *blast*
then have $x \in$ *rel-interior f* $\longleftrightarrow$ *hyperplane-equiv A z x* **for** *x*
using *relint-f zrelf* **by** *presburger*
ultimately show *?thesis*
by (*metis equalityI hyperplane-cell mem-Collect-eq subset-iff*)
qed
have *4*: *rel-interior f* $\subseteq$ *S*
by (*meson face-of-imp-subset order-trans rel-interior-subset that*(*1*))
show *?thesis*
using *1 2 3 4* **by** *blast*
qed
have *hyper2*: (*closure c face-of S* $\wedge$ *aff-dim* (*closure c*) = *d*) $\wedge$ *rel-interior* (*closure c*) = *c*
if *c*: *hyperplane-cell A c* **and** $c \subseteq S$ *aff-dim c* = *d* **for** *c*
proof (*intro conjI*)
obtain *J* **where** $J \subseteq H$ **and** J: $c = (\bigcap h \in J.\ \{x.\ (fa\ h) \cdot x < 0\}) \cap (\bigcap h \in (H - J).\ \{x.\ (fa\ h) \cdot x = 0\})$
proof −
obtain *z* **where** *z*: $c = \{y.\ \forall x \in H.\ \ sgn\ (fa\ x \cdot y) = sgn\ (fa\ x \cdot z)\}$
using *c* **by** (*force simp*: *hyperplane-cell A-def hyperplane-equiv-def hyperplane-side-def*)
show *thesis*
proof
let $?J = \{h \in H.\ sgn(fa\ h \cdot z) = -1\}$
have *1*: $fa\ h \cdot x < 0$
if $\forall h{\in}H.\ sgn\ (fa\ h \cdot x) = sgn\ (fa\ h \cdot z)$ **and** $h \in H$ **and** $sgn\ (fa\ h \cdot z) = -\ 1$ **for** *x h*
using *that* **by** (*metis sgn-1-neg*)
have *2*: $sgn\ (fa\ h \cdot z) = -\ 1$
if $\forall h{\in}H.\ sgn\ (fa\ h \cdot x) = sgn\ (fa\ h \cdot z)$ **and** $h \in H$ **and** $fa\ h \cdot x \neq 0$ **for** *x h*
proof −
have $[\![0 < fa\ h \cdot x;\ 0 < fa\ h \cdot z]\!] \Longrightarrow False$
using *that fa* **by** (*smt* (*verit*, *del-insts*) *Inter-iff Seq* ‹$c \subseteq S$› *mem-Collect-eq subset-iff z*)
then show *?thesis*
by (*metis that sgn-if sgn-zero-iff*)
qed
have *3*: $sgn\ (fa\ h \cdot x) = sgn\ (fa\ h \cdot z)$
if $h \in H$ **and** $\forall h.\ h \in H \wedge sgn\ (fa\ h \cdot z) = -\ 1 \longrightarrow fa\ h \cdot x < 0$

and $\forall h{\in}H - \{h \in H.\ sgn\ (fa\ h \cdot z) = -\ 1\}.\ fa\ h \cdot x = 0$
for *x h*
using *that 2* **by** (*metis* (*mono-tags*, *lifting*) *Diff-iff mem-Collect-eq sgn-neg*)
show $c = (\bigcap h \in ?J.\ \{x.\ fa\ h \cdot x < 0\}) \cap (\bigcap h{\in}H - ?J.\ \{x.\ fa\ h \cdot x = 0\})$
unfolding *z* **by** (*auto intro*: *1 2 3*)
qed *auto*
qed
have *finite J*
using ‹*J* ⊆ *H*› ‹*finite H*› *finite-subset* **by** *blast*
show *closure c face-of S*
proof −
have *cc*: $closure\ c = closure\ (\bigcap h{\in}J.\ \{x.\ fa\ h \cdot x < 0\}) \cap closure\ (\bigcap h{\in}H - J.\ \{x.\ fa\ h \cdot x = 0\})$
unfolding *J*
proof (*rule closure-Int-convex*)
show $convex\ (\bigcap h{\in}J.\ \{x.\ fa\ h \cdot x < 0\})$
by (*simp add*: *convex-INT convex-halfspace-lt*)
show $convex\ (\bigcap h{\in}H - J.\ \{x.\ fa\ h \cdot x = 0\})$
by (*simp add*: *convex-INT convex-hyperplane*)
have *o1*: $open\ (\bigcap h{\in}J.\ \{x.\ fa\ h \cdot x < 0\})$
by (*metis open-INT*[*OF* ‹*finite J*›] *open-halfspace-lt*)
have *o2*: $openin\ (top\text{-}of\text{-}set\ (affine\ hull\ (\bigcap h{\in}H - J.\ \{x.\ fa\ h \cdot x = 0\})))\ (\bigcap h{\in}H - J.\ \{x.\ fa\ h \cdot x = 0\})$
proof −
have $affine\ (\bigcap h{\in}H - J.\ \{n.\ fa\ h \cdot n = 0\})$
using *affine-hyperplane* **by** *auto*
then show *?thesis*
by (*metis* (*no-types*) *affine-hull-eq openin-subtopology-self*)
qed
show $rel\text{-}interior\ (\bigcap h{\in}J.\ \{x.\ fa\ h \cdot x < 0\}) \cap rel\text{-}interior\ (\bigcap h{\in}H - J.\ \{x.\ fa\ h \cdot x = 0\}) \neq \{\}$
by (*metis nonempty-hyperplane-cell c rel-interior-open o1 rel-interior-openin o2 J*)
qed
have *clo-im-J*: $closure\ `\ ((\lambda h.\ \{x.\ fa\ h \cdot x < 0\})\ `\ J) = (\lambda h.\ \{x.\ fa\ h \cdot x \le 0\})\ `\ J$
using ‹*J* ⊆ *H*› **by** (*force simp*: *image-comp fa*)
have *cleq*: $closure\ (\bigcap h{\in}H - J.\ \{x.\ fa\ h \cdot x = 0\}) = (\bigcap h{\in}H - J.\ \{x.\ fa\ h \cdot x = 0\})$
by (*intro closure-closed*) (*blast intro*: *closed-hyperplane*)
have ∗∗: $(\bigcap h{\in}J.\ \{x.\ fa\ h \cdot x \le 0\}) \cap (\bigcap h{\in}H - J.\ \{x.\ fa\ h \cdot x = 0\})$ *face-of S*
if $(\bigcap h{\in}J.\ \{x.\ fa\ h \cdot x < 0\}) \neq \{\}$
proof (*cases J*=*H*)
case *True*
have [*simp*]: $(\bigcap x{\in}H.\ \{xa.\ fa\ x \cdot xa \le 0\}) = \bigcap H$
using *fa* **by** *auto*

```
          show ?thesis
            using ‹polyhedron S› by (simp add: Seq True polyhedron-imp-convex
face-of-refl)
        next
          case False
          have **: (⋂h∈J. {n. fa h · n ≤ 0}) ∩ (⋂h∈H − J. {x. fa h · x = 0})
=
                  (⋂h∈H − J. S ∩ {x. fa h · x = 0})  (is ?L = ?R)
          proof
            show ?L ⊆ ?R
              by clarsimp (smt (verit) DiffI InterI Seq fa mem-Collect-eq)
            show ?R ⊆ ?L
              using False Seq ‹J ⊆ H› fa by blast
          qed
          show ?thesis
            unfolding **
          proof (rule face-of-Inter)
            show (λh. S ∩ {x. fa h · x = 0}) ‘ (H − J) ≠ {}
              using False ‹J ⊆ H› by blast
            show T face-of S
              if T: T ∈ (λh. S ∩ {x. fa h · x = 0}) ‘ (H − J) for T
            proof −
              obtain h where h: T = S ∩ {x. fa h · x = 0} and h ∈ H h ∉ J
                using T by auto
              have S ∩ {x. fa h · x = 0} face-of S
              proof (rule face-of-Int-supporting-hyperplane-le)
                show convex S
                  by (simp add: assms(1) polyhedron-imp-convex)
                show fa h · x ≤ 0 if x ∈ S for x
                  using that Seq fa ‹h ∈ H› by auto
              qed
              then show ?thesis
                using h by blast
            qed
          qed
        qed
        have *: ⋀S. S ∈ (λh. {x. fa h · x < 0}) ‘ J ⟹ convex S ∧ open S
          using convex-halfspace-lt open-halfspace-lt by fastforce
        show ?thesis
          unfolding cc
          apply (simp add: * closure-Inter-convex-open)
          by (metis ** cleq clo-im-J image-image)
      qed
      show aff-dim (closure c) = int d
        by (simp add: that)
      show rel-interior (closure c) = c
          by (metis ‹finite A› c convex-rel-interior-closure hyperplane-cell-convex
hyperplane-cell-relative-interior)
    qed
```

have *rel-interior* ‘ $\{f.\ f$ *face-of* $S \wedge$ *aff-dim* $f = int\ d\}$
$= \{C.$ *hyperplane-cell* $A\ C \wedge C \subseteq S \wedge$ *aff-dim* $C = int\ d\}$
using *hyper1 hyper2* **by** *fastforce*
then show *bij-betw* (*rel-interior*) $\{f.\ f$ *face-of* $S \wedge$ *aff-dim* $f = int\ d\}$ $\{C.$ *hyperplane-cell* $A\ C \wedge C \subseteq S \wedge$ *aff-dim* $C = int\ d\}$
unfolding *bij-betw-def inj-on-def* **by** (*metis* (*mono-tags*) *hyper1 mem-Collect-eq*)

qed
show *?thesis*
by (*simp add*: *Euler-characteristic* ‹*finite A*›)
qed
also have $\ldots = 0$
proof −
have A: *hyperplane-cellcomplex* A ($-\ h$) **if** $h \in H$ **for** h
proof (*rule hyperplane-cellcomplex-mono* [*OF hyperplane-cell-cellcomplex*])
have $-\ h = \{x.\ fa\ h \cdot x = 0\} \vee -\ h = \{x.\ fa\ h \cdot x < 0\} \vee -\ h = \{x.\ 0 < fa\ h \cdot x\}$
by (*smt* (*verit*, *ccfv-SIG*) *Collect-cong Collect-neg-eq fa that*)
then show *hyperplane-cell* $\{(fa\ h,0)\}$ ($-\ h$)
by (*simp add*: *hyperplane-cell-singleton fa that*)
show $\{(fa\ h,0)\} \subseteq A$
by (*simp add*: *A-def that*)
qed
then have $\bigwedge h.\ h \in H \Longrightarrow$ *hyperplane-cellcomplex* $A\ h$
using *hyperplane-cellcomplex-Compl* **by** *fastforce*
then have *hyperplane-cellcomplex* $A\ S$
by (*simp add*: *Seq hyperplane-cellcomplex-Inter*)
then have D: *Euler-characteristic* A ($UNIV$::$'n$ *set*) $=$
Euler-characteristic A ($\bigcap H$) $+$ *Euler-characteristic* A ($-\bigcap H$)
using *Euler-characteristic-cellcomplex-Un*
by (*metis Compl-partition Diff-cancel Diff-eq Seq* ‹*finite A*› *disjnt-def hyperplane-cellcomplex-Compl*)
have *Euler-characteristic A UNIV* $=$ *Euler-characteristic* $\{\}$ ($UNIV$::$'n$ *set*)
by (*simp add*: *Euler-characterstic-invariant* ‹*finite A*›)
then have E: *Euler-characteristic A UNIV* $= (-1)$ ^ ($DIM('n)$)
by (*simp add*: *Euler-characteristic-cell*)
have DD: *Euler-characteristic* A ($\bigcap$ (*uminus* ‘ J)) $= (-\ 1)$ ^ $DIM('n)$
if $J \neq \{\}$ $J \subseteq H$ **for** J
proof −
define B **where** $B \equiv (\lambda h.\ (fa\ h,0::real))$ ‘ J
then have $B \subseteq A$
by (*simp add*: *A-def image-mono that*)
have $\exists x.\ y = -x$ **if** $y \in \bigcap$ (*uminus* ‘ H) **for** y::$'n$ — Weirdly, the assumption is not used
by (*metis add.inverse-inverse*)
moreover have $-x \in \bigcap$ (*uminus* ‘ H) $\longleftrightarrow x \in$ *interior* S **for** x
proof −
have 1: *interior* $S = \{x \in S.\ \forall h \in H.\ fa\ h \cdot x < 0\}$
using *rel-interior-polyhedron-explicit* [*OF* ‹*finite H*› - *fa*]

```
        by (metis (no-types, lifting) inf-top-left Hsub Seq ‹affine hull S = UNIV›
rel-interior-interior)
      have 2: ⋀x y. ⟦y ∈ H; ∀ h∈H. fa h · x < 0; − x ∈ y⟧ ⟹ False
        by (smt (verit, best) fa inner-minus-right mem-Collect-eq)
      show ?thesis
        apply (simp add: 1)
        by (smt (verit) 2 ∗ fa Inter-iff Seq inner-minus-right mem-Collect-eq)
    qed
    ultimately have INT-Compl-H: ⋂ (uminus ‘ H) = uminus ‘ interior S
      by blast
    obtain z where z: z ∈ ⋂ (uminus ‘ J)
      using ‹J ⊆ H› ‹⋂ (uminus ‘ H) = uminus ‘ interior S› intS by fastforce
    have ⋂ (uminus ‘ J) = Collect (hyperplane-equiv B z) (is ?L = ?R)
    proof
      show ?L ⊆ ?R
        using fa ‹J ⊆ H› z
        by (fastforce simp: hyperplane-equiv-def hyperplane-side-def B-def set-eq-iff
)
      show ?R ⊆ ?L
        using z ‹J ⊆ H› apply (clarsimp simp add: hyperplane-equiv-def hyper-
plane-side-def B-def)
        by (metis fa in-mono mem-Collect-eq sgn-le-0-iff)
    qed
    then have hyper-B: hyperplane-cell B (⋂ (uminus ‘ J))
      by (metis hyperplane-cell)
    have Euler-characteristic A (⋂ (uminus ‘ J)) = Euler-characteristic B (⋂
(uminus ‘ J))
    proof (rule Euler-characterstic-invariant [OF ‹finite A›])
      show finite B
        using ‹B ⊆ A› ‹finite A› finite-subset by blast
      show hyperplane-cellcomplex A (⋂ (uminus ‘ J))
      by (meson ‹B ⊆ A› hyper-B hyperplane-cell-cellcomplex hyperplane-cellcomplex-mono)
      show hyperplane-cellcomplex B (⋂ (uminus ‘ J))
        by (simp add: hyper-B hyperplane-cell-cellcomplex)
    qed
    also have ... = (− 1) ^ nat (aff-dim (⋂ (uminus ‘ J)))
      using Euler-characteristic-cell hyper-B by blast
    also have ... = (− 1) ^ DIM('n)
    proof −
      have affine hull ⋂ (uminus ‘ H) = UNIV
          by (simp add: INT-Compl-H affine-hull-nonempty-interior intS inte-
rior-negations)
      then have affine hull ⋂ (uminus ‘ J) = UNIV
        by (metis Inf-superset-mono hull-mono subset-UNIV subset-antisym sub-
set-image-iff that(2))
      with aff-dim-eq-full show ?thesis
        by (metis nat-int)
    qed
    finally show ?thesis .
```

qed
have *EE*: $(\sum\mathcal{T} \mid \mathcal{T} \subseteq uminus\ `\ H \wedge \mathcal{T}\neq\{\}.\ (-1)\ \widehat{}\ (card\ \mathcal{T} + 1) * Euler\text{-}characteristic\ A\ (\bigcap\mathcal{T}))$
$= (\sum\mathcal{T} \mid \mathcal{T} \subseteq uminus\ `\ H \wedge \mathcal{T} \neq \{\}.\ (-1)\ \widehat{}\ (card\ \mathcal{T} + 1) * (-\ 1)\ \widehat{}\ DIM('n))$
by (*intro sum.cong* [*OF refl*]) (*fastforce simp*: *subset-image-iff intro*!: *DD*)
also have $\ldots = (-1)\ \widehat{}\ DIM('n)$
proof −
have *A*: $(\sum y = 1..card\ H.\ \sum t{\in}\{x \in \{\mathcal{T}.\ \mathcal{T} \subseteq uminus\ `\ H \wedge \mathcal{T} \neq \{\}\}.\ card\ x = y\}.\ (-\ 1)\ \widehat{}\ (card\ t + 1))$
$= (\sum\mathcal{T}{\in}\{\mathcal{T}.\ \mathcal{T} \subseteq uminus\ `\ H \wedge \mathcal{T} \neq \{\}\}.\ (-\ 1)\ \widehat{}\ (card\ \mathcal{T} + 1))$
proof (*rule sum.group*)
have $\bigwedge C.\ [\![C \subseteq uminus\ `\ H;\ C \neq \{\}]\!] \Longrightarrow Suc\ 0 \leq card\ C \wedge card\ C \leq card\ H$
by (*meson* ‹*finite H*› *card-eq-0-iff finite-surj le-zero-eq not-less-eq-eq surj-card-le*)
then show $card\ `\ \{\mathcal{T}.\ \mathcal{T} \subseteq uminus\ `\ H \wedge \mathcal{T} \neq \{\}\} \subseteq \{1..card\ H\}$
by *force*
qed (*auto simp*: ‹*finite H*›)

have $(\sum n = Suc\ 0..card\ H.\ -\ (int\ (card\ \{x.\ x \subseteq uminus\ `\ H \wedge x \neq \{\} \wedge card\ x = n\}) * (-\ 1)\ \widehat{}\ n))$
$= (\sum n = Suc\ 0..card\ H.\ (-1)\ \widehat{}\ (Suc\ n) * (card\ H\ choose\ n))$
proof (*rule sum.cong* [*OF refl*])
fix *n*
assume $n \in \{Suc\ 0..card\ H\}$
then have $\{\mathcal{T}.\ \mathcal{T} \subseteq uminus\ `\ H \wedge \mathcal{T} \neq \{\} \wedge card\ \mathcal{T} = n\} = \{\mathcal{T}.\ \mathcal{T} \subseteq uminus\ `\ H \wedge card\ \mathcal{T} = n\}$
by *auto*
then have $card\{\mathcal{T}.\ \mathcal{T} \subseteq uminus\ `\ H \wedge \mathcal{T} \neq \{\} \wedge card\ \mathcal{T} = n\} = card\ (uminus\ `\ H)\ choose\ n$
by (*simp add*: ‹*finite H*› *n-subsets*)
also have $\ldots = card\ H\ choose\ n$
by (*metis card-image double-complement inj-on-inverseI*)
finally
show $-\ (int\ (card\ \{\mathcal{T}.\ \mathcal{T} \subseteq uminus\ `\ H \wedge \mathcal{T} \neq \{\} \wedge card\ \mathcal{T} = n\}) * (-\ 1)\ \widehat{}\ n) = (-\ 1)\ \widehat{}\ Suc\ n * int\ (card\ H\ choose\ n)$
by *simp*
qed
also have $\ldots = -\ (\sum k = Suc\ 0..card\ H.\ (-1)\ \widehat{}\ k * (card\ H\ choose\ k))$
by (*simp add*: *sum-negf*)
also have $\ldots = 1 - (\sum k{=}0..card\ H.\ (-1)\ \widehat{}\ k * (card\ H\ choose\ k))$
using *atLeastSucAtMost-greaterThanAtMost* **by** (*simp add*: *sum.head* [*of 0*])
also have $\ldots = 1 - 0\ \widehat{}\ card\ H$
using *binomial-ring* [*of* -1 *1*::*int card H*] **by** (*simp add*: *mult.commute atLeast0AtMost*)
also have $\ldots = 1$
using *Seq* ‹*finite H*› ‹$S \neq UNIV$› *card-0-eq* **by** *auto*
finally have *C*: $(\sum n = Suc\ 0..card\ H.\ -\ (int\ (card\ \{x.\ x \subseteq uminus\ `\ H \wedge$

$x \neq \{\} \wedge$ *card* $x = n\}) * (-\ 1)$ ^ $n)) = (1::int)$ **.**

    have $(\sum\mathcal{T} \mid \mathcal{T} \subseteq$ *uminus* ' $H \wedge \mathcal{T} \neq \{\}.\ (-\ 1)$ ^ $($*card* $\mathcal{T} + 1)) = (1::int)$
      unfolding *A* [*symmetric*] **by** (*simp add*: *C*)
    then show *?thesis*
      by (*simp flip*: *sum-distrib-right power-Suc*)
  qed
   finally have $(\sum\mathcal{T} \mid \mathcal{T} \subseteq$ *uminus* ' $H \wedge \mathcal{T}\neq\{\}.\ (-1)$ ^ $($*card* $\mathcal{T} + 1) *$ *Euler-characteristic A* $(\bigcap\mathcal{T}))$
        $= (-1)$ ^ $DIM('n)$ **.**
  then have *Euler-characteristic A* $(\bigcup$ (*uminus* ' $H)) = (-1)$ ^ $(DIM('n))$
    using *Euler-characteristic-inclusion-exclusion* [*OF* ‹*finite A*›]
    by (*smt* (*verit*) *A Collect-cong* ‹*finite H*› *finite-imageI image-iff sum.cong*)
  then show *?thesis*
    using *D E* **by** (*simp add*: *uminus-Inf Seq*)
 qed
 finally show *?thesis* **.**
qed

7.6 Euler-Poincare relation for special $(n - 1)$-dimensional polytope

lemma *Euler-Poincare-lemma*:
 fixes *p* :: *'n::euclidean-space set*
 assumes $DIM('n) \geq 2$ *polytope p* $i \in$ *Basis* **and** *affp*: *affine hull* $p = \{x.\ x \cdot i = 1\}$
 shows $(\sum d = 0..DIM('n) - 1.\ (-1)$ ^ $d *$ *int* (*card* $\{f.\ f$ *face-of* $p \wedge$ *aff-dim* $f = $ *int* $d\})) = 1$
proof −
 have *aff-dim* $p =$ *aff-dim* $\{x.\ i \cdot x = 1\}$
  by (*metis* (*no-types*, *lifting*) *Collect-cong aff-dim-affine-hull affp inner-commute*)
 also have ... $=$ *int* $(DIM('n) - 1)$
   using *aff-dim-hyperplane* [*of i 1*] ‹$i \in$ *Basis*› **by** *fastforce*
 finally have *AP*: *aff-dim* $p =$ *int* $(DIM('n) - 1)$ **.**
 show *?thesis*
 proof (*cases* $p = \{\}$)
   case *True*
   with *AP* **show** *?thesis* **by** *simp*
 next
   case *False*
   define *S* **where** $S \equiv$ *conic hull p*
   have *1*: (*conic hull f*) $\cap \{x.\ x \cdot i = 1\} = f$ **if** $f \subseteq \{x.\ x \cdot i = 1\}$ **for** *f*
     using *that*
       by (*smt* (*verit*, *ccfv-threshold*) *affp conic-hull-Int-affine-hull hull-hull inner-zero-left mem-Collect-eq*)
   obtain *K* **where** *finite K* **and** *K*: $p =$ *convex hull K*
     by (*meson assms*(*2*) *polytope-def*)
   then have *convex-cone hull* $K =$ *conic hull* (*convex hull K*)
     using *False convex-cone-hull-separate-nonempty* **by** *auto*

then have *polyhedron S*
using *polyhedron-convex-cone-hull*
by (*simp add*: *S-def* ‹*polytope p*› *polyhedron-conic-hull-polytope*)
then have *convex S*
by (*simp add*: *polyhedron-imp-convex*)
then have *conic S*
by (*simp add*: *S-def conic-conic-hull*)
then have $0 \in S$
by (*simp add*: *False S-def*)
have $S \neq UNIV$
proof
assume $S = UNIV$
then have *conic hull p* $\cap \{x.\ x{\cdot}i = 1\} = p$
by (*metis 1 affp hull-subset*)
then have *bounded* $\{x.\ x \cdot i = 1\}$
using *S-def* ‹$S = UNIV$› *assms*(2) *polytope-imp-bounded* **by** *auto*
then obtain B **where** $B>0$ **and** B: $\bigwedge x.\ x \in \{x.\ x \cdot i = 1\} \Longrightarrow$ *norm* $x \leq B$
using *bounded-normE* **by** *blast*
define x **where** $x \equiv (\sum b{\in}Basis.\ (if\ b{=}i\ then\ 1\ else\ B{+}1) *_R b)$
obtain j **where** j: $j \in Basis\ j{\neq}i$
using ‹$DIM('n) \geq 2$›
by (*metis DIM-complex DIM-ge-Suc0 card-2-iff' card-le-Suc0-iff-eq euclidean-space-class.finite-Basis le-antisym*)
have $B{+}1 \leq |x \cdot j|$
using j **by** (*simp add*: *x-def*)
also have $\ldots \leq$ *norm x*
using *Basis-le-norm j* **by** *blast*
finally have *norm* $x > B$
by *simp*
moreover have $x \cdot i = 1$
by (*simp add*: *x-def* ‹$i \in Basis$›)
ultimately show *False*
using B **by** *force*
qed
have $S \neq \{\}$
by (*metis False S-def empty-subsetI equalityI hull-subset*)
have $\bigwedge c\ x.\ [\![0 < c;\ x \in p;\ x \neq 0]\!] \Longrightarrow 0 < (c *_R x) \cdot i$
by (*metis* (*mono-tags*) *Int-Collect Int-iff affp hull-inc inner-commute inner-scaleR-right mult.right-neutral*)
then have *doti-gt0*: $0 < x \cdot i$ **if** S: $x \in S$ **and** $x \neq 0$ **for** x
using *that* **by** (*auto simp*: *S-def conic-hull-explicit*)
have $\bigwedge a.\ \{a\}$ *face-of* $S \Longrightarrow a = 0$
using ‹*conic S*› *conic-contains-0 face-of-conic* **by** *blast*
moreover have $\{0\}$ *face-of S*
proof −
have $\bigwedge a\ b\ u.\ [\![a \in S;\ b \in S;\ a \neq b;\ u < 1;\ 0 < u;\ (1 - u) *_R a + u *_R b = 0]\!] \Longrightarrow False$
using *conic-def euclidean-all-zero-iff inner-left-distrib scaleR-eq-0-iff*
by (*smt* (*verit*, *del-insts*) *doti-gt0* ‹*conic S*› ‹$i \in Basis$›)

then show *?thesis*
by (*auto simp*: *in-segment face-of-singleton extreme-point-of-def* ‹*0* ∈ *S*›)
qed
ultimately have *face-0*: $\{f.\ f\ \mathit{face\text{-}of}\ S \wedge (\exists a.\ f = \{a\})\} = \{\{0\}\}$
by *auto*
have *interior* $S \neq \{\}$
proof
assume *interior* $S = \{\}$
then obtain *a b* **where** $a \neq 0$ **and** *ab*: $S \subseteq \{x.\ a \cdot x = b\}$
by (*metis* ‹*convex S*› *empty-interior-subset-hyperplane*)
have $\{x.\ x \cdot i = 1\} \subseteq \{x.\ a \cdot x = b\}$
by (*metis S-def ab affine-hyperplane affp hull-inc subset-eq subset-hull*)
moreover have $\neg\ \{x.\ x \cdot i = 1\} \subset \{x.\ a \cdot x = b\}$
using *aff-dim-hyperplane* [*of a b*]
by (*metis AP* ‹$a \neq 0$› *aff-dim-eq-full-gen affine-hyperplane affp hull-subset less-le-not-le subset-hull*)
ultimately have $S \subseteq \{x.\ x \cdot i = 1\}$
using *ab* **by** *auto*
with ‹$S \neq \{\}$› **show** *False*
using ‹*conic S*› *conic-contains-0* **by** *fastforce*
qed
then have $(\sum d = 0..DIM('n).\ (-1)\ \hat{}\ d * \mathit{int}\ (\mathit{card}\ \{f.\ f\ \mathit{face\text{-}of}\ S \wedge \mathit{aff\text{-}dim}\ f = \mathit{int}\ d\})) = 0$
using *Euler-polyhedral-cone* ‹$S \neq UNIV$› ‹*conic S*› ‹*polyhedron S*› **by** *blast*
then have $1 + (\sum d = 1..DIM('n).\ (-1)\ \hat{}\ d * (\mathit{card}\ \{f.\ f\ \mathit{face\text{-}of}\ S \wedge \mathit{aff\text{-}dim}\ f = d\})) = 0$
by (*simp add*: *sum.atLeast-Suc-atMost aff-dim-eq-0 face-0*)
moreover have $(\sum d = 1..DIM('n).\ (-1)\ \hat{}\ d * (\mathit{card}\ \{f.\ f\ \mathit{face\text{-}of}\ S \wedge \mathit{aff\text{-}dim}\ f = d\}))$
$= - (\sum d = 0..DIM('n) - 1.\ (-1)\ \hat{}\ d * \mathit{int}\ (\mathit{card}\ \{f.\ f\ \mathit{face\text{-}of}\ p \wedge \mathit{aff\text{-}dim}\ f = \mathit{int}\ d\}))$
proof −
have $(\sum d = 1..DIM('n).\ (-1)\ \hat{}\ d * (\mathit{card}\ \{f.\ f\ \mathit{face\text{-}of}\ S \wedge \mathit{aff\text{-}dim}\ f = d\}))$
$= (\sum d = \mathit{Suc}\ 0..\mathit{Suc}\ (DIM('n){-}1).\ (-1)\ \hat{}\ d * (\mathit{card}\ \{f.\ f\ \mathit{face\text{-}of}\ S \wedge \mathit{aff\text{-}dim}\ f = d\}))$
by *auto*
also have ... $= - (\sum d = 0..DIM('n) - 1.\ (-1)\ \hat{}\ d * \mathit{card}\ \{f.\ f\ \mathit{face\text{-}of}\ S \wedge \mathit{aff\text{-}dim}\ f = 1 + \mathit{int}\ d\})$
unfolding *sum.atLeast-Suc-atMost-Suc-shift* **by** (*simp add*: *sum-negf*)
also have ... $= - (\sum d = 0..DIM('n) - 1.\ (-1)\ \hat{}\ d * \mathit{card}\ \{f.\ f\ \mathit{face\text{-}of}\ p \wedge \mathit{aff\text{-}dim}\ f = \mathit{int}\ d\})$
proof −
{ fix *d*
assume $d \leq DIM('n) - \mathit{Suc}\ 0$
have *conic-face-p*: (*conic hull f*) *face-of S* **if** *f face-of p* **for** *f*
proof (*cases* $f=\{\}$)
case *False*
have $\{c *_R x\ |c\ x.\ 0 \leq c \wedge x \in f\} \subseteq \{c *_R x\ |c\ x.\ 0 \leq c \wedge x \in p\}$
using *face-of-imp-subset that* **by** *blast*

moreover
have *convex* $\{c *_R x \mid c\ x.\ 0 \leq c \wedge x \in f\}$
by (*metis* (*no-types*) *cone-hull-expl convex-cone-hull face-of-imp-convex that*)
moreover
have $(\exists c\ x.\ ca *_R a = c *_R x \wedge 0 \leq c \wedge x \in f) \wedge (\exists c\ x.\ cb *_R b = c *_R x \wedge 0 \leq c \wedge x \in f)$
if $\forall a{\in}p.\ \forall b{\in}p.\ (\exists x{\in}f.\ x \in$ *open-segment* $a\ b) \longrightarrow a \in f \wedge b \in f$
and $0 \leq ca\ a \in p\ 0 \leq cb\ b \in p$
and $0 \leq cx\ x \in f$ **and** *oseg*: $cx *_R x \in$ *open-segment* $(ca *_R a)\ (cb *_R b)$
for *ca a cb b cx x*
proof −
have *ai*: $a \cdot i = 1$ **and** *bi*: $b \cdot i = 1$
using *affp hull-inc that*(*3*,*5*) **by** *fastforce*+
have *xi*: $x \cdot i = 1$
using *affp that* ‹*f face-of p*› *face-of-imp-subset hull-subset* **by** *fastforce*
show *?thesis*
proof (*cases* $cx *_R x = 0$)
case *True*
then show *?thesis*
using ‹$\{0\}$ *face-of S*› *face-ofD* ‹*conic S*› *that*
by (*smt* (*verit*, *best*) *S-def conic-def hull-subset insertCI singletonD subsetD*)
next
case *False*
then have $cx \neq 0\ x \neq 0$
by *auto*
obtain *u* **where** $0 < u\ u < 1$ **and** *u*: $cx *_R x = (1 - u) *_R (ca *_R a) + u *_R (cb *_R b)$
using *oseg in-segment*(*2*) **by** *metis*
show *?thesis*
proof (*cases* $x = a$)
case *True*
then have *ua*: $(cx - (1 - u) * ca) *_R a = (u * cb) *_R b$
using *u* **by** (*simp add*: *algebra-simps*)
then have $(cx - (1 - u) * ca) * 1 = u * cb * 1$
by (*metis ai bi inner-scaleR-left*)
then have $a{=}b \vee cb = 0$
using *ua* ‹$0 < u$› **by** *force*
then show *?thesis*
by (*metis True scaleR-zero-left that*(*2*) *that*(*4*) *that*(*7*))
next
case *False*
show *?thesis*
proof (*cases* $x = b$)
case *True*
then have *ub*: $(cx - (u * cb)) *_R b = ((1 - u) * ca) *_R a$
using *u* **by** (*simp add*: *algebra-simps*)

```
        then have (cx − (u ∗ cb)) ∗ 1 = ((1 − u) ∗ ca) ∗ 1
          by (metis ai bi inner-scaleR-left)
        then have a=b ∨ ca = 0
          using ‹u < 1› ub by auto
        then show ?thesis
          using False True that(4) that(7) by auto
      next
        case False
        have cx > 0
          using ‹cx ≠ 0› ‹0 ≤ cx› by linarith
        have False if ca = 0
        proof −
          have cx = u ∗ cb
      by (metis add-0 bi inner-real-def inner-scaleR-left real-inner-1-right
scale-eq-0-iff that u xi)
          then show False
            using ‹x ≠ b› ‹cx ≠ 0› that u by force
        qed
        with ‹0 ≤ ca› have ca > 0
          by force
        have aff: x ∈ affine hull p ∧ a ∈ affine hull p ∧ b ∈ affine hull p
          using affp xi ai bi by blast
        show ?thesis
        proof (cases cb=0)
          case True
          have u′: cx ∗R x = ((1 − u) ∗ ca) ∗R a
            using u by (simp add: True)
          then have cx = ((1 − u) ∗ ca)
            by (metis ai inner-scaleR-left mult.right-neutral xi)
          then show ?thesis
            using True u′ ‹cx ≠ 0› ‹ca ≥ 0› ‹x ∈ f› by auto
        next
          case False
          with ‹cb ≥ 0› have cb > 0
            by linarith
          { have False if a=b
            proof −
              have ∗: cx ∗R x = ((1 − u) ∗ ca + u ∗ cb) ∗R b
                using u that by (simp add: algebra-simps)
              then have cx = ((1 − u) ∗ ca + u ∗ cb)
                by (metis xi bi inner-scaleR-left mult.right-neutral)
              with ‹x ≠ b› ‹cx ≠ 0› ∗ show False
                by force
            qed
          }
          moreover
           have cx ∗R x /R cx = (((1 − u) ∗ ca) ∗R a + (cb ∗ u) ∗R b)
/R cx
            using u by simp
```

```
              then have xeq: x = ((1−u) ∗ ca / cx) ∗R a + (cb ∗ u / cx) ∗R b
            by (simp add: ‹cx ≠ 0› divide-inverse-commute scaleR-right-distrib)
                then have proj: 1 = ((1−u) ∗ ca / cx) + (cb ∗ u / cx)
                  using ai bi xi by (simp add: inner-left-distrib)
                then have eq: cx + ca ∗ u = ca + cb ∗ u
                  using ‹cx > 0› by (simp add: field-simps)
                have ∃ u>0. u < 1 ∧ x = (1 − u) ∗R a + u ∗R b
                proof (intro exI conjI)
                  show 0 < inverse cx ∗ u ∗ cb
                    by (simp add: ‹0 < cb› ‹0 < cx› ‹0 < u›)
                  show inverse cx ∗ u ∗ cb < 1
                        using proj ‹0 < ca› ‹0 < cx› ‹u < 1› by (simp add:
divide-simps)
                  show x = (1 − inverse cx ∗ u ∗ cb) ∗R a + (inverse cx ∗ u ∗
cb) ∗R b
                    using eq ‹cx ≠ 0› by (simp add: xeq field-simps)
                qed
                ultimately show ?thesis
                  using that by (metis in-segment(2))
              qed
            qed
          qed
        qed
      qed
      ultimately show ?thesis
        using that by (auto simp: S-def conic-hull-explicit face-of-def)
    qed auto
    moreover
    have conic-hyperplane-eq: conic hull (f ∩ {x. x · i = 1}) = f
      if f face-of S 0 < aff-dim f for f
    proof
      show conic hull (f ∩ {x. x · i = 1}) ⊆ f
        by (metis ‹conic S› face-of-conic inf-le1 subset-hull that(1))
      have ∃ c x'. x = c ∗R x' ∧ 0 ≤ c ∧ x' ∈ f ∧ x' · i = 1 if x ∈ f for x
      proof (cases x=0)
        case True
        obtain y where y ∈ f y ≠ 0
            by (metis ‹0 < aff-dim f› aff-dim-sing aff-dim-subset insertCI
linorder-not-le subset-iff)
        then have y · i > 0
          using ‹f face-of S› doti-gt0 face-of-imp-subset by blast
        then have y /R (y · i) ∈ f ∧ (y /R (y · i)) · i = 1
        using ‹conic S› ‹f face-of S› ‹y ∈ f› conic-def face-of-conic by fastforce
        then show ?thesis
          using True by fastforce
      next
        case False
        then have x · i > 0
          using ‹f face-of S› doti-gt0 face-of-imp-subset that by blast
```

then have $x /_R (x \cdot i) \in f \wedge (x /_R (x \cdot i)) \cdot i = 1$
using ‹*conic S*› ‹*f face-of S*› ‹$x \in f$› *conic-def face-of-conic* **by** *fastforce*
then show *?thesis*
by (*metis* ‹$0 < x \cdot i$› *divideR-right eucl-less-le-not-le*)
qed
then show $f \subseteq$ *conic hull* $(f \cap \{x.\ x \cdot i = 1\})$
by (*auto simp*: *conic-hull-explicit*)
qed

have *conic-face-S*: *conic hull f face-of S*
if *f face-of S* **for** *f*
by (*metis* ‹*conic S*› *face-of-conic hull-same that*)

have *aff-1d*: *aff-dim* (*conic hull f*) = *aff-dim f* + *1* (**is** *?lhs* = *?rhs*)
if *f face-of p* **and** $f \neq \{\}$ **for** *f*
proof (*rule order-antisym*)
have *?lhs* $\leq$ *aff-dim*(*affine hull* (*insert 0* (*affine hull f*)))
proof (*intro aff-dim-subset hull-minimal*)
show $f \subseteq$ *affine hull insert 0* (*affine hull f*)
by (*metis hull-insert hull-subset insert-subset*)
show *conic* (*affine hull insert 0* (*affine hull f*))
by (*metis affine-hull-span-0 conic-span hull-inc insertI1*)
qed
also have ... $\leq$ *?rhs*
by (*simp add*: *aff-dim-insert*)
finally show *?lhs* $\leq$ *?rhs* **.**
have *aff-dim f* < *aff-dim* (*conic hull f*)
proof (*intro aff-dim-psubset psubsetI*)
show *affine hull f* $\subseteq$ *affine hull* (*conic hull f*)
by (*simp add*: *hull-mono hull-subset*)
have $0 \notin$ *affine hull f*
using *affp face-of-imp-subset hull-mono that*(*1*) **by** *fastforce*
moreover have $0 \in$ *affine hull* (*conic hull f*)
by (*simp add*: ‹$f \neq \{\}$› *hull-inc*)
ultimately show *affine hull f* $\neq$ *affine hull* (*conic hull f*)
by *auto*
qed
then show *?rhs* $\leq$ *?lhs*
by *simp*
qed

have *face-S-imp-face-p*: $\bigwedge f.$ *f face-of S* $\Longrightarrow$ $f \cap \{x.\ x \cdot i = 1\}$ *face-of p*
by (*metis 1 S-def affp convex-affine-hull face-of-slice hull-subset*)

have *conic-eq-f*: *conic hull f* $\cap \{x.\ x \cdot i = 1\} = f$
if *f face-of p* **for** *f*
by (*metis 1 affp face-of-imp-subset hull-subset le-inf-iff that*)

have *dim-f-hyperplane*: *aff-dim* $(f \cap \{x.\ x \cdot i = 1\})$ = *int d*

if *f face-of S aff-dim f* = *1* + *int d* **for** *f*
proof −
have *conic f*
using ‹*conic S*› *face-of-conic that*(*1*) **by** *blast*
then have *0* ∈ *f*
using *conic-contains-0 that* **by** *force*
moreover have ¬ *f* ⊆ {*0*}
using *subset-singletonD that*(*2*) **by** *fastforce*
ultimately obtain *y* **where** *y*: *y* ∈ *f y* ≠ *0*
by *blast*
then have *y* · *i* > *0*
using *doti-gt0 face-of-imp-subset that*(*1*) **by** *blast*
have *aff-dim* (*conic hull* (*f* ∩ {*x*. *x* · *i* = *1*})) = *aff-dim* (*f* ∩ {*x*. *x* · *i* = *1*}) + *1*
proof (*rule aff-1d*)
show *f* ∩ {*x*. *x* · *i* = *1*} *face-of p*
by (*simp add*: *face-S-imp-face-p that*(*1*))
have *inverse*(*y* · *i*) $*_R$ *y* ∈ *f*
using ‹*0* < *y* · *i*› ‹*conic S*› *conic-mul face-of-conic that*(*1*) *y*(*1*) **by** *fastforce*
moreover have *inverse*(*y* · *i*) $*_R$ *y* ∈ {*x*. *x* · *i* = *1*}
using ‹*y* · *i* > *0*› **by** (*simp add*: *field-simps*)
ultimately show *f* ∩ {*x*. *x* · *i* = *1*} ≠ {}
by *blast*
qed
then show *?thesis*
by (*simp add*: *conic-hyperplane-eq that*)
qed
have *card* {*f*. *f face-of S* ∧ *aff-dim f* = *1* + *int d*}
= *card* {*f*. *f face-of p* ∧ *aff-dim f* = *int d*}
proof (*intro bij-betw-same-card bij-betw-imageI*)
show *inj-on* (λ*f*. *f* ∩ {*x*. *x* · *i* = *1*}) {*f*. *f face-of S* ∧ *aff-dim f* = *1* + *int d*}
by (*smt* (*verit*) *conic-hyperplane-eq inj-on-def mem-Collect-eq of-nat-less-0-iff*)

show (λ*f*. *f* ∩ {*x*. *x* · *i* = *1*}) ' {*f*. *f face-of S* ∧ *aff-dim f* = *1* + *int d*} = {*f*. *f face-of p* ∧ *aff-dim f* = *int d*}
using *aff-1d conic-eq-f conic-face-p*
by (*fastforce simp*: *image-iff face-S-imp-face-p dim-f-hyperplane*)
qed
}
then show *?thesis*
by *force*
qed
finally show *?thesis* **.**
qed
ultimately show *?thesis*
by *auto*
qed

qed

corollary *Euler-poincare-special*:
fixes $p :: 'n::euclidean\text{-}space\ set$
assumes $2 \leq DIM('n)$ *polytope* p $i \in Basis$ **and** *affp*: *affine hull* $p = \{x.\ x \cdot i = 0\}$
shows $(\sum d = 0..DIM('n) - 1.\ (-1)\ \widehat{}\ d * card\ \{f.\ f\ face\text{-}of\ p \wedge aff\text{-}dim\ f = d\}) = 1$
proof −
{ fix d
have *eq*: $image((+)\ i)\ `\ \{f.\ f\ face\text{-}of\ p\} \cap image((+)\ i)\ `\ \{f.\ aff\text{-}dim\ f = int\ d\} = image((+)\ i)\ `\ \{f.\ f\ face\text{-}of\ p\} \cap \{f.\ aff\text{-}dim\ f = int\ d\}$
by (*auto simp*: *aff-dim-translation-eq*)
have $card\ \{f.\ f\ face\text{-}of\ p \wedge aff\text{-}dim\ f = int\ d\} = card\ (image((+)\ i)\ `\ \{f.\ f\ face\text{-}of\ p \wedge aff\text{-}dim\ f = int\ d\})$
by (*simp add*: *inj-on-image card-image*)
also have $\ldots = card\ (image((+)\ i)\ `\ \{f.\ f\ face\text{-}of\ p\} \cap \{f.\ aff\text{-}dim\ f = int\ d\})$
by (*simp add*: *Collect-conj-eq image-Int inj-on-image eq*)
also have $\ldots = card\ \{f.\ f\ face\text{-}of\ (+)\ i\ `\ p \wedge aff\text{-}dim\ f = int\ d\}$
by (*simp add*: *Collect-conj-eq faces-of-translation*)
finally have $card\ \{f.\ f\ face\text{-}of\ p \wedge aff\text{-}dim\ f = int\ d\} = card\ \{f.\ f\ face\text{-}of\ (+)\ i\ `\ p \wedge aff\text{-}dim\ f = int\ d\}$ **.**
}
then
have $(\sum d = 0..DIM('n) - 1.\ (-1)\ \widehat{}\ d * card\ \{f.\ f\ face\text{-}of\ p \wedge aff\text{-}dim\ f = d\}) = (\sum d = 0..DIM('n) - 1.\ (-1)\ \widehat{}\ d * card\ \{f.\ f\ face\text{-}of\ (+)\ i\ `\ p \wedge aff\text{-}dim\ f = int\ d\})$
by *simp*
also have $\ldots = 1$
proof (*rule Euler-Poincare-lemma*)
have $\bigwedge x.\ [\![i \in Basis;\ x \cdot i = 1]\!] \Longrightarrow \exists y.\ y \cdot i = 0 \wedge x = y + i$
by (*metis add-cancel-left-left eq-diff-eq inner-diff-left inner-same-Basis*)
then show *affine hull* $(+)\ i\ `\ p = \{x.\ x \cdot i = 1\}$
using ‹$i \in Basis$› **unfolding** *affine-hull-translation affp* **by** (*auto simp*: *algebra-simps*)
qed (*use assms polytope-translation-eq* **in** *auto*)
finally show *?thesis* **.**
qed

7.7 Now Euler-Poincare for a general full-dimensional polytope

theorem *Euler-Poincare-full*:
fixes $p :: 'n::euclidean\text{-}space\ set$
assumes *polytope* p $aff\text{-}dim\ p = DIM('n)$
shows $(\sum d = 0..DIM('n).\ (-1)\ \widehat{}\ d * (card\ \{f.\ f\ face\text{-}of\ p \wedge aff\text{-}dim\ f = d\})) = 1$
proof −

define *augm*:: $'n \Rightarrow 'n \times$ *real* **where** *augm* $\equiv \lambda x.\ (x,0)$
define *S* **where** $S \equiv$ *augm* ‘ *p*
obtain i::$'n$ **where** i: $i \in$ *Basis*
by (*meson SOME-Basis*)
have *bounded-linear augm*
by (*auto simp*: *augm-def bounded-linearI'*)
then have *polytope S*
unfolding *S-def* **using** *polytope-linear-image* ‹*polytope p*› *bounded-linear.linear*
by *blast*
have *face-pS*: $\bigwedge F$. *F face-of p* $\longleftrightarrow$ *augm* ‘ *F face-of S*
using *S-def* ‹*bounded-linear augm*› *augm-def bounded-linear.linear face-of-linear-image inj-on-def* **by** *blast*
have *aff-dim-eq*[*simp*]: *aff-dim* (*augm* ‘ *F*) = *aff-dim F* **for** *F*
using ‹*bounded-linear augm*› *aff-dim-injective-linear-image bounded-linear.linear*

unfolding *augm-def inj-on-def* **by** *blast*
have $*$: {*F*. *F face-of S* $\wedge$ *aff-dim F* = *int d*} = (*image augm*) ‘ {*F*. *F face-of p* $\wedge$ *aff-dim F* = *int d*}
(**is** *?lhs* = *?rhs*) **for** *d*
proof
have $\bigwedge G$. ⟦*G face-of S*; *aff-dim G* = *int d*⟧
$\Longrightarrow \exists F$. *F face-of p* $\wedge$ *aff-dim F* = *int d* $\wedge$ *G* = *augm* ‘ *F*
by (*metis face-pS S-def aff-dim-eq face-of-imp-subset subset-imageE*)
then show *?lhs* $\subseteq$ *?rhs*
by (*auto simp*: *image-iff*)
qed (*auto simp*: *image-iff face-pS*)
have *ceqc*: *card* {*f*. *f face-of S* $\wedge$ *aff-dim f* = *int d*} = *card* {*f*. *f face-of p* $\wedge$ *aff-dim f* = *int d*} **for** *d*
unfolding $*$
by (*rule card-image*) (*auto simp*: *inj-on-def augm-def*)
have ($\sum d = 0..DIM('n \times real) - 1.\ (- 1)$ ^ $d * $ *int* (*card* {*f*. *f face-of S* $\wedge$ *aff-dim f* = *int d*})) = 1
proof (*rule Euler-poincare-special*)
show $2 \leq DIM('n \times real)$
by *auto*
have *snd0*: $(a, b) \in$ *affine hull S* $\Longrightarrow b = 0$ **for** *a b*
using *S-def* ‹*bounded-linear augm*› *affine-hull-linear-image augm-def* **by** *blast*
moreover have $\bigwedge a.\ (a, 0) \in$ *affine hull S*
using *S-def* ‹*bounded-linear augm*› *aff-dim-eq-full affine-hull-linear-image assms*(2) *augm-def* **by** *blast*
ultimately show *affine hull S* = $\{x.\ x \cdot (0::'n, 1::real) = 0\}$
by *auto*
qed (*auto simp*: ‹*polytope S*› *Basis-prod-def*)
then show *?thesis*
by (*simp add*: *ceqc*)
qed

In particular, the Euler relation in 3 dimensions

corollary *Euler-relation*:

fixes p :: $'n$::*euclidean-space set*
assumes *polytope p aff-dim p* = *3 DIM*($'n$) = *3*
shows (*card* {*v. v face-of p* ∧ *aff-dim v* = *0*} + *card* {*f. f face-of p* ∧ *aff-dim f* = *2*}) − *card* {*e. e face-of p* ∧ *aff-dim e* = *1*} = *2*
proof −
have ⋀*x*. ⟦*x face-of p*; *aff-dim x* = *3*⟧ ⟹ *x* = *p*
using *assms* **by** (*metis face-of-aff-dim-lt less-irrefl polytope-imp-convex*)
then have *3*: {*f. f face-of p* ∧ *aff-dim f* = *3*} = {*p*}
using *assms* **by** (*auto simp*: *face-of-refl polytope-imp-convex*)
have (∑ *d* = *0..3*. (−*1*) ^ *d* ∗ *int* (*card* {*f. f face-of p* ∧ *aff-dim f* = *int d*})) = *1*
using *Euler-Poincare-full* [*of p*] *assms* **by** *simp*
then show *?thesis*
by (*simp add*: *sum.atLeast0-atMost-Suc-shift numeral-3-eq-3 3*)
qed

end

References

[1] I. Lakatos. *Proofs and Refutations: The Logic of Mathematical Discovery*. 1976.

[2] J. Lawrence. A short proof of Euler's relation for convex polytopes. *Canadian Mathematical Bulletin*, 40(4):471–474, 1997.